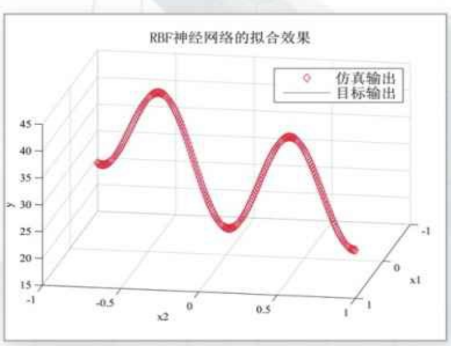
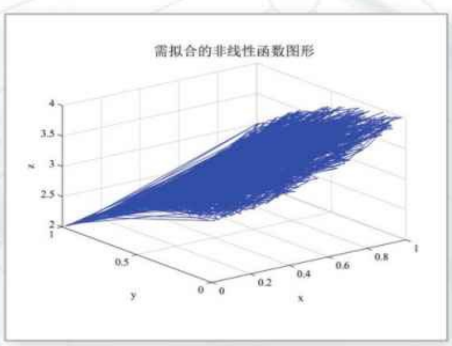
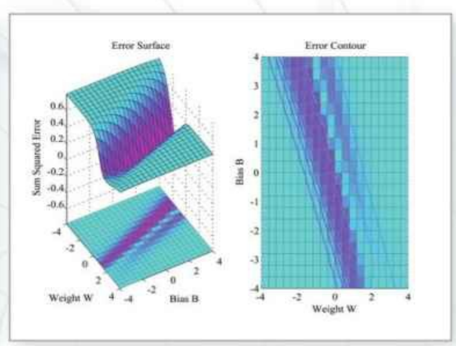
工程软件应用精解

MATLAB神经网络超级学习手册

MATLAB技术联盟 刘冰 郭海霞 编著







人民邮电出版社
POSTS & TELECOM PRESS

目　录

参考文献

工程软件应用精解

MATLAB神经网络超级学习手册

MATLAB技术联盟　刘冰　郭海霞　编著

人民邮电出版社

北京

图书在版编目（**CIP**）数据

MATLAB神经网络超级学习手册/刘冰，郭海霞编著.--北京：人民邮电出版社，2014.5

ISBN　978-7-115-34948-4

Ⅰ.①M…　Ⅱ.①刘…②郭…　Ⅲ.①人工神经网络—Matlab软件—手册　Ⅳ.①TP183-62

中国版本图书馆CIP数据核字（2014）第061424号

内容提要

本书以新近推出的 MATLAB R2013a 神经网络工具箱为基础，系统全面地介绍了神经网络的各种概念和应用。本书按逻辑编排，自始至终采用实例描述；内容完整且每章相对独立，是一本不可多得的掌握MATLAB神经网络的学习用书。

全书共分为16章，从MATLAB简介开始，详细介绍了MATLAB的基础知识、MATLAB程序设计、人工神经网络概述、感知器、线性神经网络、BP神经网络、RBF神经网络、反馈型神经网络、竞争型神经网络、神经网络在 Simulink 中的应用、神经网络 GUI、自定义神经网络及函数等内容。在本书最后，还详细介绍了神经网络在MATLAB中的几种应用方法。

本书以神经网络结构为主线，以学习算法为副线，结合各种实例，目的是使读者易看懂、会应用。本书是一本简明介绍MATLAB神经网络设计技能的综合性用书。本书深入浅出，实例引导，讲解详实，既可以作为高等院校理工科的研究生、本科生的教材，也可作为广大科研工程技术人员的参考用书。

◆编著　MATLAB技术联盟　刘冰　郭海霞

责任编辑　王峰松

责任印制　彭志环　杨林杰

◆人民邮电出版社出版发行　　北京市丰台区成寿寺路11号

邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷
◆开本：787×1092　1/16
印张：30
字数：709千字　　2014年5月第1版
印数：1-30000册　　2014年5月北京第1次印刷
定价：69.00元
读者服务热线：（**010**）**81055410**　印装质量热线：（**010**）**81055316**
反盗版热线：（**010**）**81055315**
广告经营许可证：京崇工商广字第**0021**号

前言

MATLAB是国际上公认的最优秀的数值计算和仿真分析软件之一，并且具有很强的开放性。到目前为止，MATLAB平台已经推出了几十种针对不同应用的工具箱，其中为满足神经网络研究而开发的神经网络工具箱是其重要组成部分。

人工神经网络（Artificial Neural Network，ANN），是一种模仿生物神经网络的结构和功能的数学模型或计算模型。神经网络由大量的人工神经元联结进行计算。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。大多数情况下，人工神经网络能在外界信息的基础上改变内部结构，是一种自适应系统。

神经网络的研究可以分为理论研究和应用研究两大方面。在网络模型与算法研究的基础上，利用人工神经网络组成实际的应用系统，例如，完成某种信号处理或模式识别的功能，构建专家系统等。

本书基于MATLAB R2013a 软件详细讲解进行神经网络设计的方法和技巧，同时给出了MATLAB在神经网络方面的应用。

1.本书特点

由浅入深，循序渐进：本书以初中级读者为对象，首先从MATLAB使用基础讲起，再以神经网络在MATLAB中的应用案例帮助读者尽快掌握神经网络设计的技能。

步骤详尽，内容新颖：本书结合作者多年MATLAB神经网络使用经验与实际工程应用案例，将MATLAB软件的使用方法与技巧详细地

讲解给读者。本书在讲解过程中步骤详尽、内容新颖，讲解过程辅以相应的图片，使读者在阅读时一目了然，从而快速把握书中所讲内容。

实例典型，轻松易学：学习实际工程应用案例的具体操作是掌握神经网络设计最好的方式。本书通过综合应用案例，透彻详尽地讲解了神经网络在各方面的应用。

2.本书内容

本书基于MATLAB R2013a 版软件，讲解了神经网络设计的基础知识和核心内容。本书主要分为两个部分：MATLAB基础知识和神经网络工具箱。

第1章 本章简单介绍了MATLAB 的基本知识，包括简介、特点、安装、工作环境等。对MATLAB软件的基本用途和方法也做了简单介绍。

第2章 本章对Simulink 做了简单介绍，包括simulink 基础以及Simulink 组成、工作原理和环境设置等。

第3章 本章介绍了MATLAB程序设计，包括MATLAB的程序结构、MATLAB 控制语句、数据的输入和输出、文件操作、程序优化和程序调试等内容。

第4章 本章介绍了人工神经网络，包括人工神经网络的基本概念、研究方向等。之后，介绍了神经网络的基本单元——神经元，并对神经网络结构和学习做了讲解，最后介绍了神经网络工具箱。

第5章 本章介绍了感知器，包括感知器的概念、学习规则、局限性和工具箱的函数等，还对感知器在MATLAB中的仿真设计做了叙述。

第6章 本章介绍了线性神经网络的应用，包括线性神经网络原理、模型、工具箱函数、基本的设计方法及其设计案例。

第7章 本章主要介绍BP 神经网络原理，包括BP 神经网络的模型、算法、训练和功能等知识，还介绍了BP神经网络的设计方法、工具箱函数及典型工程应用案例。

第8章 本章主要介绍RBF 神经网络模型，包含其工作原理和具体实现，还介绍了RBF神经网络的学习算法、工具箱函数、非线性滤波及RBF在MATLAB中的应用实例。

第9章 本章主要介绍反馈型神经网络，包括反馈型神经网络的基本概念、Hopfield网络模型、Hopfield网络工具箱函数、离散型Hopfield网络和连续型Hopfield网络等，最后还对Elman网络进行了说明。

第10章 本章主要介绍竞争型神经网络，包括自组织型竞争神经网络、自组织特征映射神经网络、自适应共振理论神经网络、学习向量量化神经网络、竞争型神经网络工具箱函数和竞争型神经网络的应用等。

第11章 本章主要介绍神经网络在Simulink中的应用，包括Simulink中的神经网络模块、神经网络模型预测控制系统、反馈线性化控制系统和模型参考控制系统等。

第12章 本章主要介绍神经网络GUI 方面的知识，包括GUI 简介、常规神经网络GUI、神经网络拟合GUI、模式识别GUI和聚类GUI，最后还对GUI的数据操作进行了介绍。

第13章 本章主要介绍自定义神经网络的创建、网络初始化、训练和仿真等内容。因为自定义函数学习难度较大，本章只是简单介绍了自定义函数的种类。

第14章 本章主要介绍随机神经网络的基本思想，模拟退火算法和Boltzmann机算法，并举例用随机神经网络解决实际TSP问题。

第15章 本章主要运用简单的实例介绍了简单的神经网络应用，包括感知器神经网络、线性神经网络、BP神经网络和RBF神经网络的使用。

第16章 本章主要介绍BP 神经网络、PID神经网络控制、遗传算法优化、模糊神经网络和概率神经网络分类预测等5个典型的神经网络应用案例。

注：本书中用到的所有程序代码和数据，请到作者博客下载。

3.读者对象

本书适合于MATLAB神经网络设计初学者和期望提高神经网络工程应用能力的读者，具体说明如下。

★ 相关从业人员　　★ 初学MATLAB神经网络的技术人员

★ 大中专院校的教师和在校生　★ 相关培训机构的教师和学员

★ 参加工作实习的“菜鸟”　★ 广大科研工作人员

4.本书作者

本书由MATLAB技术联盟刘冰、郭海霞编写，另外孔玲军、李昕、刘成柱、史洁玉、孙国强、代晶、贺碧蛟、石良臣、柯维娜等人为本书的编写提供了大量的帮助，在此一并表示感谢。

虽然作者在本书的编写过程中力求叙述准确、完善，但由于水平有限，书中难免存在错误和欠妥之处，希望读者和同仁能够及时指出，共同促进本书质量的提高。

5.读者服务

读者朋友在学习过程中遇到与本书有关的技术问题，可以发邮件到邮箱book_hai@126.com，或者访问博客 http://blog.sina.com.cn/tecbook，我们会尽快给予解答，并竭诚为您服务。

编者

2013年秋

第1章 MATLAB简介

本章主要介绍MATLAB软件的基本用途和方法。MATLAB是目前在国际上被广泛接受和使用的科学与工程计算软件。虽然Cleve Moler教授开发它的初衷是为了更简单、更快捷地解决矩阵运算，但现在的MATLAB已经发展成为一种集数值运算、符号运算、数据可视化、图形界面设计、程序设计、仿真等多种功能于一体的集成软件。

本章将介绍MATLAB的发展、特点、安装、工作环境和MATLAB帮助系统，力图使读者能初步熟悉MATLAB软件的基本知识。

学习目标：

·了解MATLAB 的发展、特点和功能

·了解MATLAB 工具箱的概念和类型

·熟练掌握MATLAB 窗口的用途和使用方法

1.1 MATLAB的发展

20 世纪 70 年代中后期，曾在密西根大学、斯坦福大学和新墨西哥大学担任数学与计算机科学教授的 Cleve Moler 博士，为讲授矩阵理论和数值分析课程的需要，和同事用Fortran语言编写了两个子程序库EISPACK和LINPACK，这便是构思和开发MATLAB的起点。MATLAB一词是对Matrix Laboratory（矩阵实验室）的缩写，由此可看出MATLAB与矩阵计算的渊源。

MATLAB除了利用EISPACK和LINPACK两大软件包的子程序外，还包含用Fortran语言编写的用于承担命令翻译的部分。

为进一步推动 MATLAB 的应用，在 20 世纪 80 年代初，John Little等人将先前的MATLAB 全部用C 语言进行改写，形成了新一代的MATLAB。1984 年，Cleve Moler和John Little等人成立MathWorks公司，并于同年向市场推出了MATLAB的第一个商业版本。

随着市场接受度的提高，其功能也不断增强，在完成数值计算的基础上，新增了数据可视化以及与其他流行软件的接口等功能，并开始了对MATLAB工具箱的研究开发。

1993年，MathWorks公司推出了基于PC的以Windows为操作系统平台的MATLAB 4.0版。

1994年推出的4.2版，扩充了4.0版的功能，尤其在图形界面设计方面提供了新的方法。

1997 年推出的MATLAB 5.0 增加了更多的数据结构，如结构数组、细胞数组、多维数组、对象、类等，使其成为一种更方便的编程语言。

1999 年初推出的MATLAB 5.3 在很多方面又进一步改进了

MATLAB 的功能。

2000年10 月底推出的全新的 MATLAB 6.0 正式版（Release 12），在核心数值算法、界面设计、外部接口、应用桌面等诸多方面有了极大的改进。

2002年8 月推出的MATLAB 6.5，其操作界面进一步集成化，并开始运用JIT 加速技术，使运算速度有了明显提高。

2004年7 月，MathWorks公司推出了MATLAB 7.0（Release l4），其中集成了MATLAB 7.0 编译器、Simulink 6.0图形仿真器及很多工具箱，在编程环境、代码效率、数据可视化、文件I/O等方面都进行了升级。

最近的一次版本更新是在2013年3月，Mathworks公司推出了MATLAB R2013a 版。该版本的新特点是引入Fixed-Point Designer，它结合Fixed-Point Toolbox、Simulink Fixed Point、 Phased Array System Toolbox 和 SimRF 的功能，增强了无线和雷达通信系统设计。R2013a还更新了80种其他产品，包括Polyspace嵌入式软件验证产品。

现在的MATLAB已经不再是仅仅解决矩阵与数值计算的软件，更是一种集数值与符号运算、数据可视化图形表示与图形界面设计、程序设计、仿真等多种功能于一体的集成软件。

观察由欧美引进的新版教材，MATLAB已经成为线性代数、数值分析计算、数学建模、信号与系统分析、自动控制、数字信号处理、通信系统仿真等一批课程的基本教学工具。而在国内，随着MATLAB在我国高校的推广和应用，MATLAB已经渐入人心。

1.2 MATLAB的特点及应用领域

MATLAB有两种基本的数据运算量：数组和矩阵，二者单从形式上不好区分。每一个变量可能被当作数组，也可能被当作矩阵，这要依靠所采用的运算法则或运算函数来定。

在MATLAB中，数组与矩阵的运算法则和运算函数是有区别的。但不论是MATLAB的数组还是MATLAB的矩阵，都已经改变了一般高级语言中使用数组的方式和解决矩阵问题的方法。

在MATLAB中，矩阵运算是把矩阵视为一个整体来进行，基本上与线性代数的处理方法一致。矩阵的加减乘除、乘方开方、指数对数等运算，都有一套专门的运算符或运算函数。

对于数组，不论是算术的运算，还是关系或逻辑的运算，甚至于调用函数的运算，形式上可以当作整体，有一套有别于矩阵的、完整的运算符和运算函数，但实质上却是针对数组的每个元素施行的。

当MATLAB把矩阵（或数组）当作一个独立的运算量来对待后，向下可以兼容向量和标量。不仅如此，矩阵和数组中的元素可以用复数作基本单元，向下可以包含实数集。这些是 MATLAB 区别于其他高级语言的根本特点。以此为基础，还可以概括出如下一些MATLAB的特色。

1.语言简洁，编程效率高

因为MATLAB定义了专门用于矩阵运算的运算符，使得矩阵运算就像列出算式执行标量运算一样简单，而且这些运算符本身就能执行向量和标量的多种运算。

利用这些运算符可使一般高级语言中的循环结构变成一个简单的MATLAB语句，再结合MATLAB丰富的库函数可使程序变得相当简

短，几条语句即可代替数十行C语言或Fortran语言程序语句的功能。

2.交互性好，使用方便

使用C语言或Fortran语言的时候，首先需要编写源程序，然后对其进行编译、连接，待形成可执行文件后，方可运行程序得出结果。

而在MATLAB的命令窗口中，输入一条命令，立即就能看到该命令的执行结果，体现了良好的交互性。交互方式减少了编程和调试程序的工作量，给使用者带来了极大的方便。

3.强大的绘图能力，便于数据可视化

MATLAB不仅能绘制多种不同坐标系中的二维曲线，还能绘制三维曲面，体现了强大的绘图能力。正是这种能力为数据的图形化表示（即数据可视化）提供了有力工具，使数据的展示更加形象生动，有利于揭示数据间的内在关系。

4.领域广泛的工具箱，便于众多学科直接使用

MATLAB工具箱（函数库）可分为两类：功能性工具箱和学科性工具箱。功能性工具箱主要用来扩充其符号计算功能、图示建模仿真功能、文字处理功能以及与硬件实时交互的功能。而学科性工具箱是专业性比较强的工具箱，如优化工具箱、统计工具箱、控制工具箱、通信工具箱、图像处理工具箱、小波工具箱等。

5.开放性好，便于扩展

除内部函数外， MATLAB 的其他文件都是公开的、可读可改的源文件，体现了MATLAB的开放性特点。使用者可修改源文件和加入自己的文件，甚至构造自己的工具箱。

6.文件I/O和外部引用程序接口

支持读入更大的文本文件，支持压缩格式的MAT文件，用户可以动态加载、删除或者重载Java类等。正是由于以上几个特点，MATLAB的应用领域十分广阔，典型的应用举例如下。

·数据分析

·数值与符号计算
·工程与科学绘图
·控制系统设计
·航天工业
·生物医学工程
·汽车工业
·语音处理
·图像与数字信号处理
·财务、金融分析
·建模、仿真及样机开发
·新算法研究开发

1.3 MATLAB R2013a 的安装

MATLAB是一个功能强大的数学工具软件，只有在适当的系统环境中才能正常运行。本节介绍MATLAB R2013a 在PC 的Windows 操作系统中的典型安装方法。

提示：在本书的后面章节中，如果没有特别说明，所有的操作都是在 Windows XP操作系统下的MATLAB R2013a中进行的。

在一般情况下，当使用者将MATLAB R2013a 的安装光盘插入光驱后，会自动启动“安装向导”。如果向导没有自动启动，可以打开安装光盘中的 setup.exe 应用程序，启动安装向导。

在安装过程中出现的所有界面都是标准界面。用户只需要按照界面中的提示进行操作，输入用户名、单位名以及软件产品序列号等。下面介绍MATLAB R2013a 安装过程中的几个主要界面。

首先，使用者需要在安装界面中选择“自定义”选项才能自行选择软件安装的组件和目录等，如图1-1所示。

当选择“自定义”选项后，单击界面中的“下一步”按钮，就会出现MATLAB安装路径选择界面，如图1-2所示。使用者在选择MATLAB软件的安装位置和软件名称的时候，可以使用任意名称，同时也可以将MATLAB软件安装在硬盘中的任何位置。

建议：在安装路径名称中不要使用中文和空格符号，避免与其他软件可能出现的不兼容问题。



图1-1 安装类型选项



图1-2 MATLAB安装路径选择界面

选择安装路径后继续单击“下一步”按钮，出现图 1-3 所示的产品选

择界面，使用者可以在该界面中选择需要安装的组件。

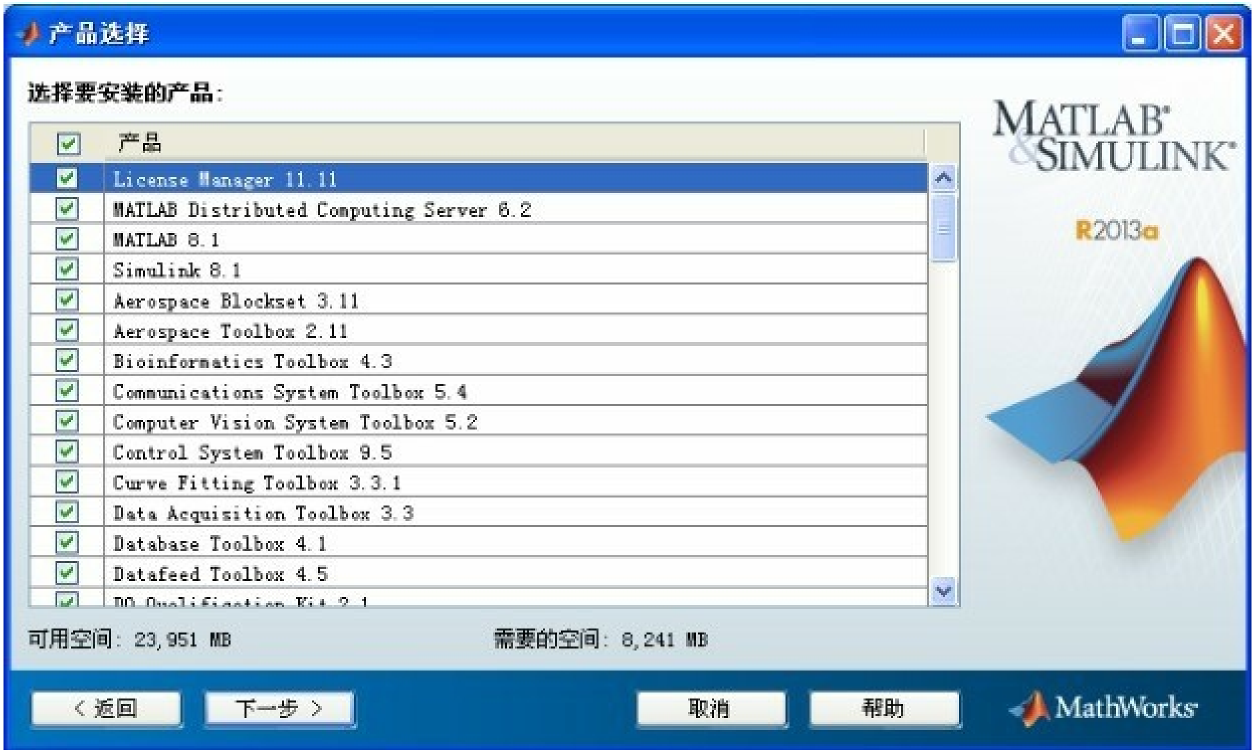


图1-3 MATLAB组件选择界面

在MATLAB组件选择界面下面的选框中，用户需要选择MATLAB软件的安装组件。可以直接勾选相应的组件，进行安装。默认情况下，系统会选中所有的组件，但是对于一般用户而言，很多组件的专业性过强，没有安装的必要。

为了帮助用户了解组件的构成，下面给出比较典型的组合方式，见表1-1。

表1-1 MATLAB安装组件

组件名称	说　　明
MATLAB	必要组件，用户需要选择该组件后才能正常安装其他部分。该组件是整个软件的核心部分。为整个软件系统提供 MATLAB 工作环境
Simulink	MATLAB 的通用性组件，可以完成 MATLAB 的常见功能。建议用户安装这些组件
Symtalic Math	
Optimization	
MATLAB Complier	
Control System	MATLAB 的常用专业组件，建议用户根据需要选择组件

当用户选择MATLAB的安装组件后，单击组件选择界面中的“下一步”按钮，就会出现确认安装界面，如图1-4（a）所示。

如果用户单击“安装”按钮，则MATLAB软件进入安装程序，如图1-4（b）所示。



（a） MATLAB 选项界面　　（b） MATLAB 安装界面

图1-4 MATLAB组件的确认及安装

1.4 MATLAB R2013a的工作环境

在一般情况下，可以使用两种方法来打开 MATLAB R2013a。在上节中，若已将快捷方式添加到桌面上，则可以双击桌面上的快捷方式图标，打开操作界面。

如果用户没有添加MATLAB快捷方式，则需要在MATLAB的安装文件夹里（默认路径为C:\Program Files\MATLAB\R2013a\bin\win32）选择MATLAB.exe 应用程序，同样可以打开MATLAB操作界面。这两种方法的结果完全相同。

1.4.1 操作界面简介

MATLAB R2013a 延续了MATLAB R2012b 的操作界面。该操作界面中包含大量的交互式界面。例如通用操作界面、工具包专业界面、帮助界面和演示界面等。这些交互性界面组合在一起，构成MALTAB的默认操作界面。

安装后首次启动MATLAB所得的操作界面如图1-5所示，这是系统默认的、未曾被用户依据自身需要和喜好设置过的界面。

MATLAB的主界面是一个高度集成的工作环境，有4个不同职责分工的窗口。它们分别是 Command Window（命令窗口）、Command History（历史命令窗口）、Current Directory （当前目录窗口）和Workspace（工作空间窗口）。

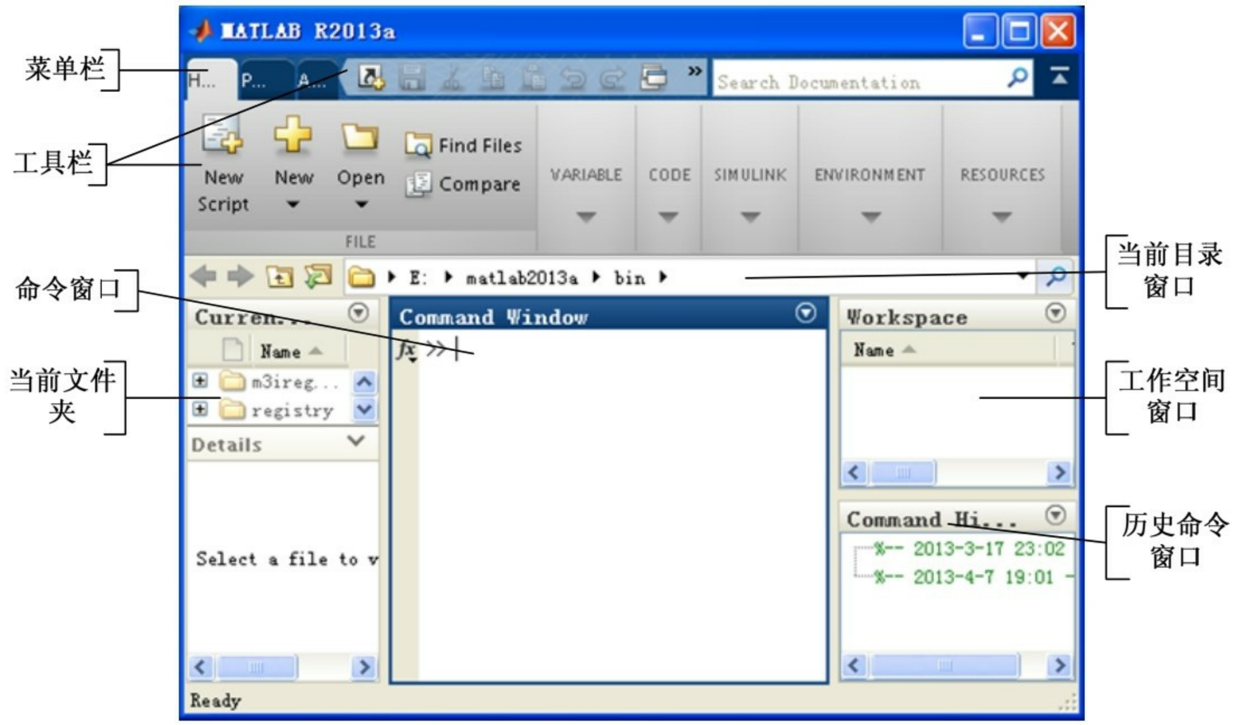


图1-5 MATLAB默认的主界面

菜单栏和工具栏在组成方式和内容上与一般应用软件基本相同，本章不再赘述。

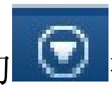
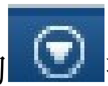
1.4.2 Workspace（命令窗口）

在MATLAB默认主界面的中间是命令窗口。因为MATLAB至今未被汉化，所有窗口名都用英文表示，所以“Command Window”即指命令窗口。

命令窗口顾名思义是接收命令输入的窗口，但实际上，可输入的对象除MATLAB命令之外，还包括函数、表达式、语句以及M文件名或MEX文件名等，为叙述方便，本书后续部分将这些可输入的对象统称为语句。

MATLAB的工作方式之一是在命令窗口中输入语句，然后由MATLAB逐句解释执行并在命令窗口中给出结果。命令窗口可显示除图形以外的所有运算结果。命令窗口可从MATLAB主界面中分离出

来，以便单独显示和操作，当然也可重新返回主界面中，其他窗口也有相同的行为。

分离命令窗口可以单击命令窗口右上角的按钮，在出现的下拉菜单中选择Undock选项，另外还可以直接用鼠标将命令窗口拖离主界面，其结果如图 1-6 所示。若将命令窗口返回到主界面中，可单击命令窗口右上角的按钮，在出现的下拉菜单中选择Dock选项。下面分几点对使用命令窗口的一些相关问题加以说明。

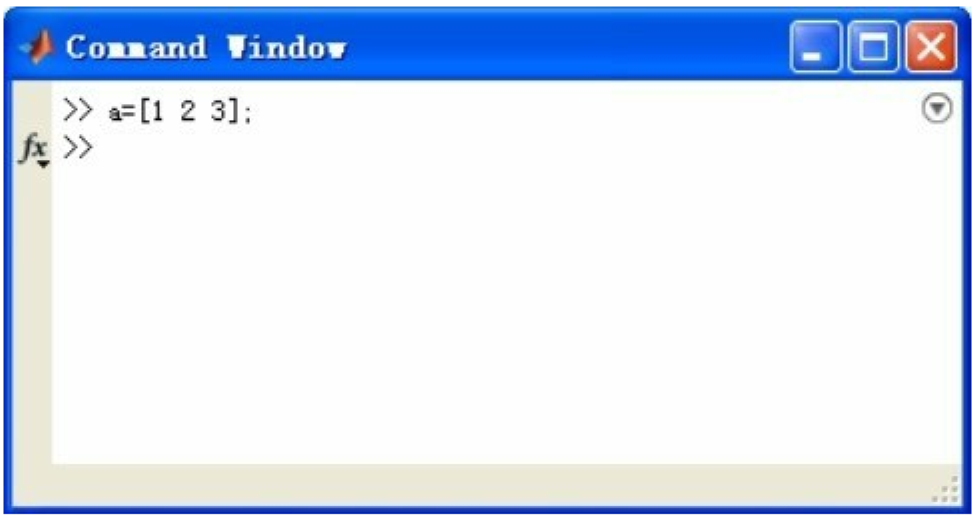


图1-6 分离的命令窗口

1.命令提示符和语句颜色

在分离的命令窗口中，每行语句前都有一个符号“>>”，此即命令提示符。在此符号后（也只能在此符号后）输入各种语句并按Enter键，方可被MATLAB接收和执行。执行的结果通常直接显示在语句下方。

不同类型语句用不同颜色区分。在默认情况下，输入的命令、函数、表达式以及计算结果等采用黑色字体，字符串采用红色，if、for等关键词采用蓝色，注释语句用绿色。

2.语句的重复调用、编辑和重运行

命令窗口不仅能编辑和运行当前输入的语句，而且对曾经输入的语句也有快捷的方法进行重复调用、编辑和运行。成功实施重复调用的前

提是已输入的语句仍然保存在历史命令窗口中（未对该窗口执行清除操作）。而重复调用和编辑的快捷方法就是利用表1-2所列的键盘按键。

表1-2 语句行用到的编辑键

键盘按键	键的用途	键盘按键	键的用途
↑	向上回调以前输入的语句行	Home	让光标跳到当前行的开头
↓	向下回调以前输入的语句行	End	让光标跳到当前行的末尾
←	光标在当前行中左移一字符	Delete	删除当前行光标后的字符
→	光标在当前行中右移一字符	Backspace	删除当前行光标前的字符

其实这些按键与文字处理软件中介绍的同一编辑键在功能上是大体一致的，不同点主要是：在文字处理软件中是针对整个文档使用，而MATLAB命令窗口是以行为单位使用这些编辑键，类似于编辑DOS命令的使用手法。

3.语句行中使用的标点符号

MATLAB 在输入语句时，可能要用到表 1-3 所列的各种符号，这些符号在 MATLAB中所起的作用见表1-3。

提示：在向命令窗口输入语句时，一定要在英文输入状态下输入，尤其在刚刚输完汉字后初学者很容易忽视中英文输入状态的切换。

表1-3 MATLAB语句中常用标点符号的作用

名 称	符 号	作 用
空格		变量分隔符；矩阵一行中各元素间的分隔符；程序语句关键词分隔符
逗号	,	分隔欲显示计算结果的各语句；变量分隔符；矩阵一行中各元素间的分隔符
点号	.	数值中的小数点；结构数组的域访问符
分号	;	分隔不想显示计算结果的各语句；矩阵行与行的分隔符
冒号	:	用于生成一维数值数组；表示一维数组的全部元素或多维数组某一维的全部元素
百分号	%	注释语句说明符，凡在其后的字符视为注释性内容而不被执行

续表

名 称	符 号	作 用
单引号	' '	字符串标识符
圆括号	( )	用于矩阵元素引用；用于函数输入变量列表；确定运算的先后次序
方括号	[ ]	向量和矩阵标识符；用于函数输出列表
花括号	{ }	标识细胞数组
续行号	…	长命令行需分行时连接下行用
赋值号	＝	将表达式赋值给一个变量

4.命令窗口中数值的显示格式

为了适应用户以不同格式显示计算结果的需要，MATLAB设计了多种数值显示格式以供用户选用，见表1-4。其中默认的显示格式是：数值为整数时，以整数显示；数值为实数时，以short格式显示；如果数值的有效数字超出了这一范围，则以科学计数法显示结果。

表1-4 命令窗口中数据e的显示格式

格 式	命令窗口中的显示形式	格式效果说明
short(默认)	2.7183	保留 4 位小数，整数部分超过 3 位的小数用 short e 格式
short e	2.7183e+000	用 1 位整数和 4 位小数表示，倍数关系用科学计数法表示成十进制指数形式
short g	2.7183	保证 5 位有效数字，数字大小在 10 的正负 5 次幂之间时，自动调整数位多少，超出幂次范围时用 short e 格式
long	2.71828182845905	14 位小数，最多 2 位整数，共 16 位十进制数，否则用 long e 格式表示
long e	2.718281828459046e+000	15 位小数的科学计数法表示
long g	2.71828182845905	保证 15 位有效数字，数字大小在 10 的+15 和–5 次幂之间时，自动调整数位多少，超出幂次范围时用 long e 格式
rational	1457/536	用分数有理数近似表示
hex	4005bf0a8b14576a	十六进制表示
+	+	正、负数和零分别用＋、－、空格表示
bank	2.72	限两位小数，用于表示元、角、分
compact	不留空行显示	在显示结果之间没有空行的压缩格式
loose	留空行显示	在显示结果之间有空行的稀疏格式

需要说明的是，表中最后2个是用于控制屏幕显示格式的，而非数值显示格式。MATLAB所有数值均按IEEE浮点标准所规定的长型格式存储，显示的精度并不代表数值实际的存储精度，或者说数值参与运算

的精度，认清这点是非常必要的。

5.数值显示格式的设定方法

格式设定的方法有两种：一是执行MATLAB工作栏中Preferences命令，用弹出的对话框去设定；二是执行 format 命令，例如要用 long 格式，在命令窗口中输入 format long语句即可。两种方法均可独立完成设定，但使用命令可方便在程序设计时进行格式设定。

不仅数值显示格式可由用户自行设置，数字和文字的字体显示风格、大小、颜色也可由用户自行挑选。其方法还是执行Preferences命令，弹出图1-7所示对话框。利用该对话框左侧的格式对象树，从中选择要设定的对象再配合相应的选项，便可对所选对象的风格、大小、颜色等进行设定。

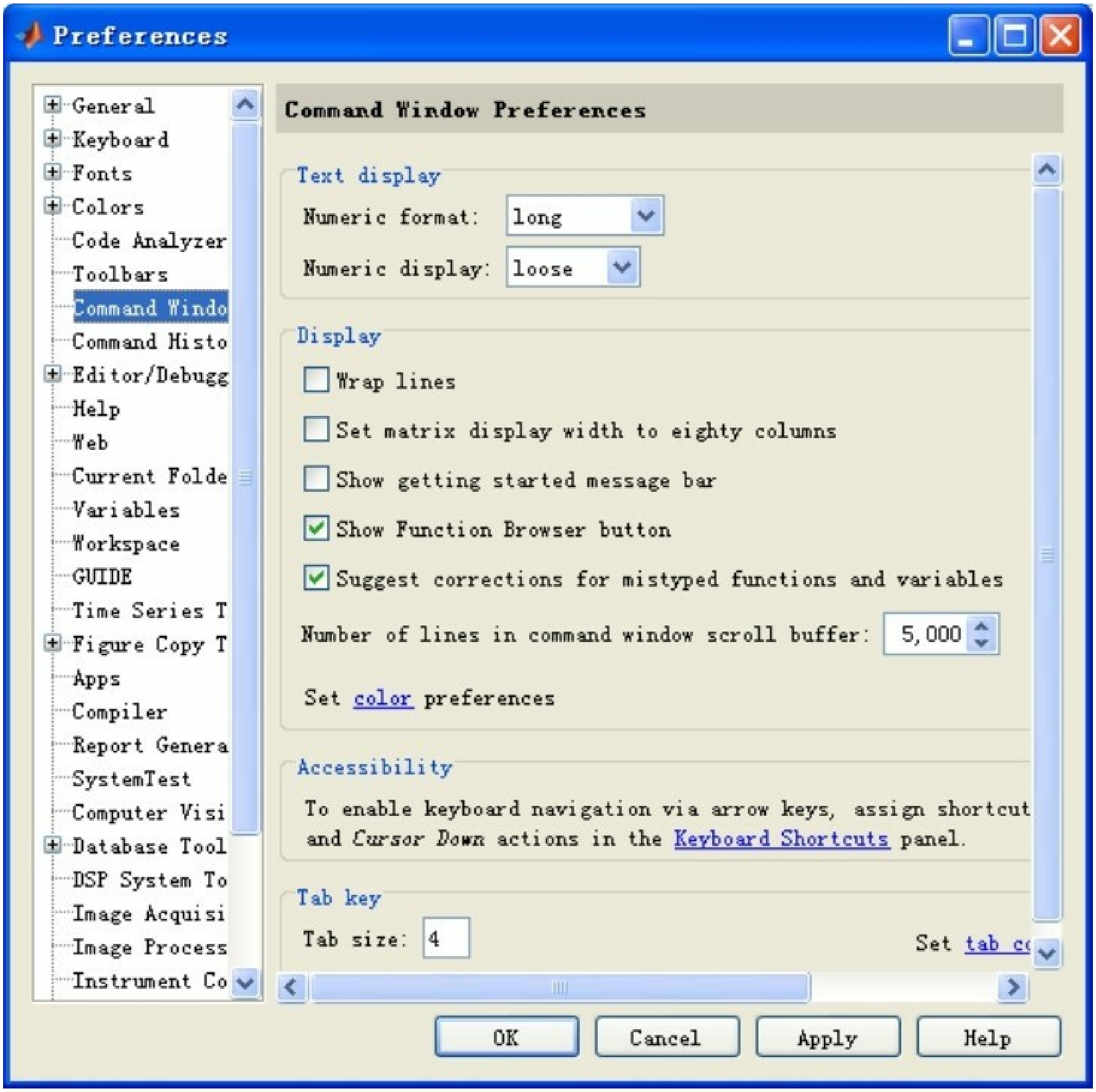


图1-7 Preferences设置对话框

6.命令窗口清屏

当命令窗口中执行过许多命令后，窗口会被占满，为方便阅读，清除屏幕显示是经常采用的操作。清除命令窗口显示通常有两种方法。

（1）执行MATLAB 工具栏的Clear Command→Command Window命令。

（2）在提示符后直接输入clc语句。

两种方法都能清除命令窗口中的显示内容，也仅仅是命令窗口的显示内容而已，并不能清除工作空间和历史命令窗口的显示内容。

1.4.3 Command History（历史命令窗口）

Command History（历史命令窗口）是MATLAB用来存放曾在命令窗口中使用过的语句的。它借用计算机的存储器来保存信息。其主要目的是为了便于用户追溯、查找曾经用过的语句，利用这些既有的资源节省编程时间。

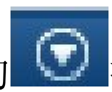
单击历史命令窗口右上角的按钮，在出现的下拉菜单中选择Undock选项，可以将其从MATLAB主界面分离出来。另外还可以直接用鼠标将历史命令窗口拖离主界面，其结果如图 1-8 所示。从窗口中记录的时间来看，其中存放的正是曾经使用过的语句。



图1-8 分离的历史命令窗口

对历史命令窗口中的内容，可在选中的前提下，将它们复制到当前正在工作的命令窗口中，以供进一步修改或直接运行。其优势在如下两种情况下体现得尤为明显：一是需要重复处理长语句；二是在选择多行曾经用过的语句形成M文件时。

提示：如果只需要执行历史命令窗口中的单条语句，那么双击历史命令窗口中的该条语句即可。

1.复制、执行历史命令窗口中的命令

历史命令窗口的主要应用体现在表1-5中。表中操作方法一栏中提到的“选中”操作，与Windows选中文件时方法相同，同样可以结合Ctrl键和Shift键使用。

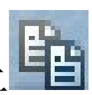
表1-5 历史命令窗口的主要应用

功　能	操作方法
复制单行或多行语句	选中单行或多行语句，执行 Edit 菜单的 Copy 命令，回到命令窗口，执行粘贴操作，即可实现复制
执行单行或多行语句	选中单行或多行语句，右击，弹出快捷菜单，执行该菜单中的 Evaluate Selection 命令，则选中语句将在命令窗口中运行，并给出相应结果。或者双击选择的语句行也可运行
把多行语句写成 M 文件	选中单行或多行语句，右击，弹出快捷菜单，执行该菜单的 Create M-File 命令，利用随之打开的 M 文件编辑/调试器窗口，可将选中语句保存为 M 文件

用历史命令窗口完成所选语句的复制操作的步骤如下。

（1）用鼠标选中所需第一行。

（2）再按Shift键和鼠标选择所需最后一行，于是连续多行即被选中。

（3）单击工具栏上图标，或在选中区域单击鼠标右键，执行快捷菜单的Copy命令。

（4）回到命令窗口，在该窗口用快捷菜单中的Paste命令，所选内容即被复制到命令窗口。其操作如图1-9所示。

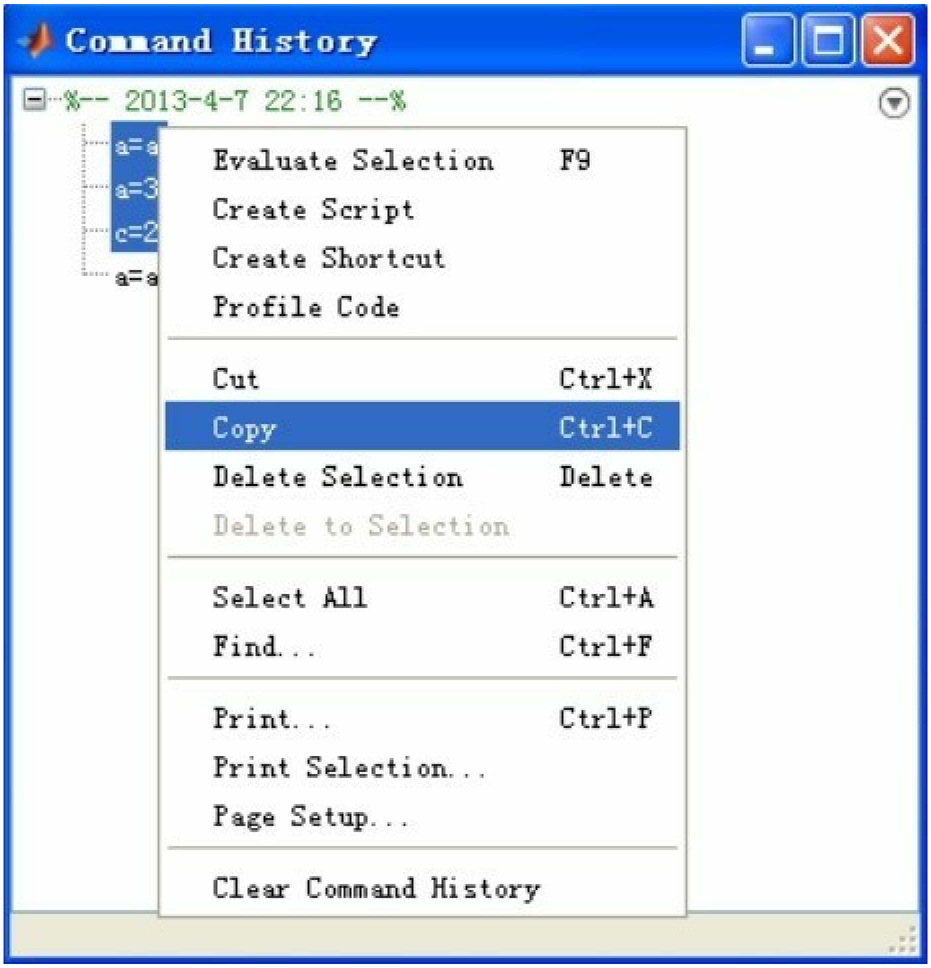


图1-9 历史命令窗口选中与复制操作

用历史命令窗口完成所选语句的运行操作的步骤如下。

（1）用鼠标选中所需第一行。

（2）再按Ctrl键结合鼠标点选所需的行，于是不连续多行即被选中。

（3）在选中的区域右击弹出快捷菜单，选用Evaluate Selection 命令，计算结果就会出现在命令窗口中。

2.清除历史命令窗口中的内容。

清除历史命令窗口内容的方法就是执行 MATLAB 工具栏中的 Clear Command→Command History命令。

提示：当执行上述命令后，历史命令窗口当前的内容就被完全清除了，以前的命令再不能被追溯和利用。

1.4.4 输入变量

在MATLAB的计算和编程过程中，变量和表达式都是最基础的元素。如果用户需要深入学习MATLAB，十分有必要了解MATLAB关于定义变量和表达式的基本规定。

在MATLAB中，为变量定义名称需要满足下列规则。

（1）变量名称和函数名称有大小写区别。对于变量名称Mu和mu，MATLAB会认为是不同的变量。exp是MATLAB内置的指数函数名称。因此，如果用户输入exp(0)，系统会得出结果 1；而如果用户输入EXP(0)，MATLAB 会显示错误的提示信息“Undefined function 'EXP' for input arguments of type 'double'.”，表明MATLAB无法识别EXP的函数名称，同时系统会提示是否需要输入exp(0)，如图1-10所示。

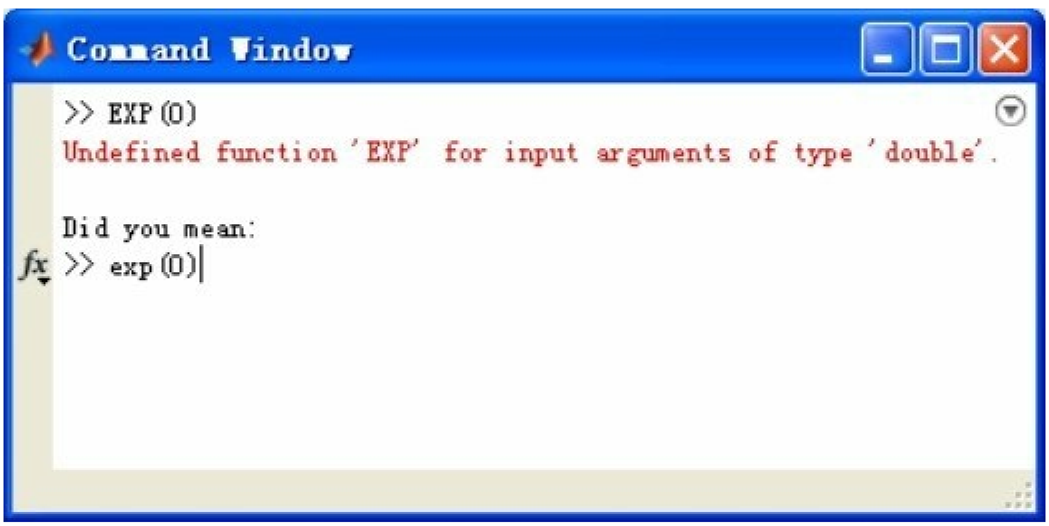


图1-10 函数名称区别大小写

（2）变量名称的第一个字符必须是英文字符。在MATLAB 6.5 以后的版本中，变量名称最多可以包含63个字符。因此，变量5xf、_mat等都是不合法的变量名称。

（3）变量名称中不可以包含空格或者标点符号。但是可以包括下划线。因此，变量名称xf_mat是合法的。

尽管MATLAB对于变量名称的限制较少，但还是建议用户在设置变量名称时考虑变量的含义。例如在M文件中，变量名称Outputname比名称a更好理解。

在上面的变量名称规则中，没有限制用户使用MATLAB的预定义变量名称，但是根据笔者的经验，还是建议用户不要使用MATLAB预先定义的变量名称。因为，用户每次启动MATLAB，系统就会自动产生这些变量，表1-6中列出了常见的预定义变量名称。

表1-6 MATLAB中的预定义变量

预定义变量	含　　义
ans	计算结果的默认名称
eps	计算机的零阈值
inf（Inf）	无穷大
Pi	圆周率
NaN（nan）	表示结果或者变量不是数值

1.4.5 路径管理

MATLAB 借鉴 Windows 资源管理器管理磁盘、文件夹和文件的思想，设计了当前文件夹窗口。利用该窗口可组织、管理和使用所有MATLAB文件和非MATLAB文件，例如新建、复制、删除和重命名文件夹和文件。甚至还可用此窗口打开、编辑和运行M程序文件以及载入MAT数据文件等。当然，其核心功能还是设置当前文件夹。

Current Folder（当前文件夹窗口）如图1-11 所示。下面主要介绍当前文件夹的概念及如何完成对当前文件夹的设置，并不准备在此讨论程序文件的运行。

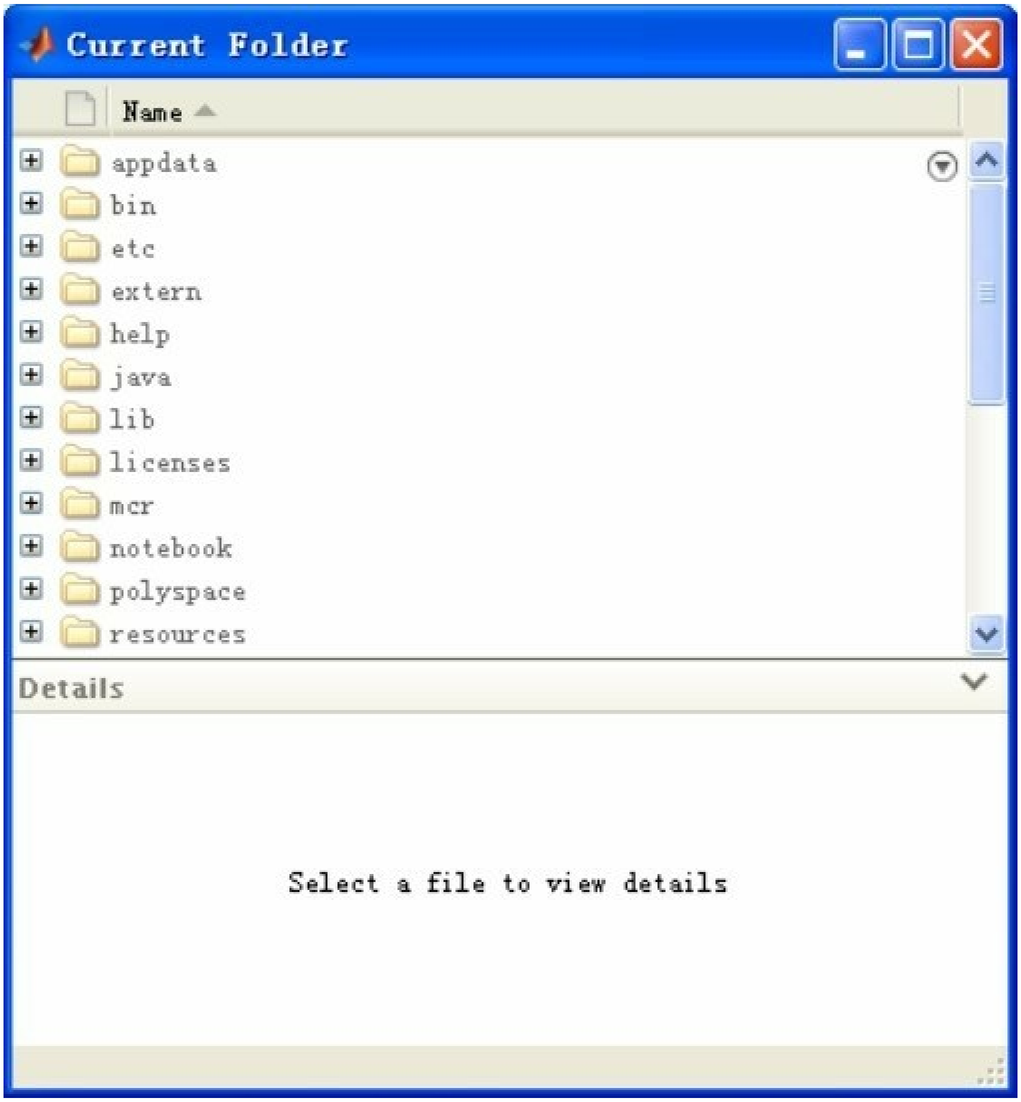


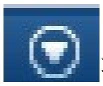
图1-11 分离的当前文件夹窗口

MATLAB的当前文件夹即是系统默认的实施打开、装载、编辑和保存文件等操作时的文件夹。用桌面图标启动MATLAB后，系统默认的当前目录是 ...\MATLAB\work。

设置当前文件夹就是将此默认文件夹改变成用户希望使用的文件夹，它应是用户提前建立好的用来存放文件和数据的文件夹。

具体的设置方法有如下两种。

（1）在当前目录设置区设置。MATLAB主界面工具栏的右边以及分离的当前目录设置窗口都有当前目录设置区，可以在设置区的下拉列

表文本框中直接填写待设置的文件夹名或选择下拉列表中已有的文件夹名。如果需要分离当前文件夹窗口，单击当前文件夹窗口右上角的按钮，在出现的下拉菜单中选择Undock选项即可。

（2）用命令设置。有一组从DOS中借用的文件夹命令可以完成这一任务，它们的语法格式见表1-7。

表1-7 几个常用的设置当前文件夹的命令

目录命令	含　义	示　例
cd	显示当前目录	cd
cd 文件夹名	设定当前目录为“文件夹名”	cd f:\matfiles

用命令设置当前文件夹，为在程序中控制当前目录的改变提供了方便，因为编写完成的程序通常用M文件存放，执行这些文件时是不便先退出再用窗口菜单或对话框去改变当前目录设置的。

1.4.6 搜索路径

MATLAB中大量的函数和工具箱文件是组织在硬盘的不同文件夹中的。用户建立的数据文件、命令和函数文件也是由用户存放在指定的文件夹中。当需要调用这些函数或文件时，找到这些函数或文件所存放的文件夹就成为首要问题，路径的概念也就因此而产生了。

路径其实就是给出存放某个待查函数和文件的文件夹名称。当然，这个文件夹名称应包括盘符和一级级嵌套的子文件夹名。

例如，现有一文件t04_06.m存放在D盘“MATLAB文件”文件夹下的“M文件”子文件夹下的“第 4 章”子文件夹中，那么，描述它的路径是：D:\MATLAB 文件\M 文件\第4章。若要调用这个M文件，可在命令窗口或程序中将其表达为：D:\MATLAB文件\M文件\第4章\t04_06.m。

在使用时，这种书写因为过长，很不方便，MATLAB为解决这一问题，引入了搜索路径机制。设置搜索路径机制就是将一些可能要被用

到的函数或文件的存放路径提前通知系统，而无须在执行和调用这些函数和文件时输入一长串的路径。

提示：必须指出，不是说有了搜索路径，MATLAB对程序中出现的符号就只能从搜索路径中去查找。在 MATLAB 中，一个符号出现在程序语句里或命令窗口的语句中可能有多种解读，它也许是一个变量、特殊常量、函数名、M文件或MEX文件等，具体将其识别成什么，涉及搜索顺序的问题。

如果在命令提示符“>>”后输入符号xt，或程序语句中有一个符号xt，那么，MATLAB将试图按下列次序去搜索和识别。

（1）在MATLAB内存中进行检查搜索，看xt是否为工作空间窗口的变量或特殊常量，如果是，则将其当成变量或特殊常量来处理，不再往下展开搜索识别。

（2）上一步否定后，检查xt是否为MATLAB的内部函数，若肯定，则调用xt这个内部函数。

（3）上一步否定后，继续在当前文件夹中搜索是否有名为“xt.m”或“xt.mex”的文件存在，若肯定，则将xt作为文件调用。

（4）上一步否定后，继续在MATLAB搜索路径的所有目录中搜索是否有名为“xt.m”或“xt.mex”的文件存在，若肯定，则将xt作为文件调用。

（5）上述4步全走完后，仍未发现xt这一符号的出处，则MATLAB发出错误信息。必须指出的是，这种搜索是以花费更多执行时间为代价的。

MATLAB设置搜索路径的方法有两种：一种是用菜单和对话框；另一种是用命令。现将两方案分述如下。

1.用菜单和对话框设置搜索路径

在MATLAB主界面的File 菜单中有Set Path 命令，执行这一命令将打开设置搜索路径的对话框，如图1-12所示。

对话框左边设计了多个按钮，其中最上面的两个按钮分别是Add Folder…和 Add with Subfolders…，单击任何一个按钮都会弹出一个名为浏览文件夹的对话框，如图1-13所示。利用浏览文件夹对话框可以从树形目录结构中选择欲指定为搜索路径的文件夹。

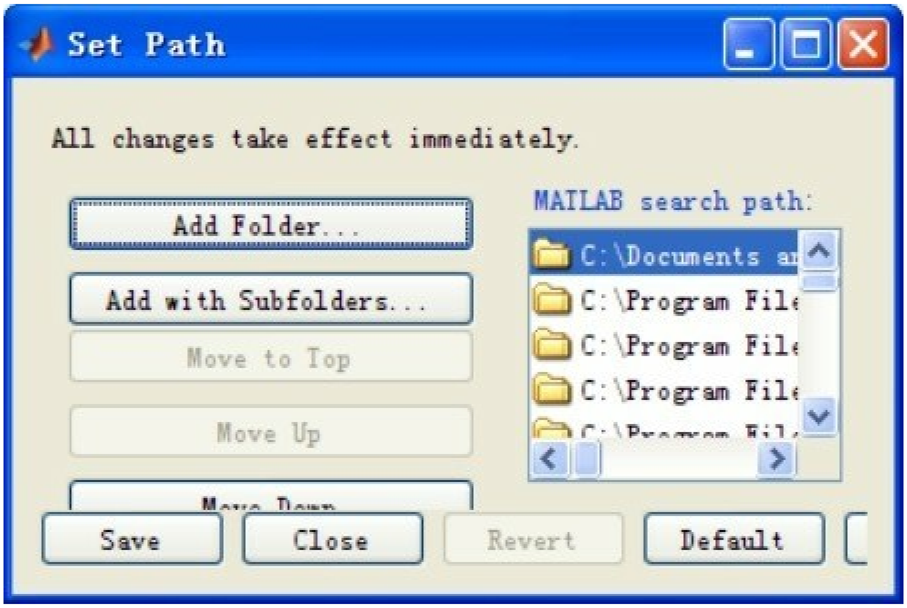


图1-12 设置搜索路径对话框



图1-13 浏览文件夹对话框

Add Folder…和Add with Subfolders…两个按钮的不同处在于后者设

置某个文件夹成为可搜索的路径后，其下级子文件夹将自动被加入到搜索路径中。

设置搜索路径对话框下面有两个按钮Save和Close，在使用时值得注意。Save按钮是用来保存对当前搜索路径所做修改的，通常先执行Save命令后，再执行Close。Close按钮是用来关闭对话框的，但是如果只想将修改过的路径为本次打开 MATLAB 使用，无意供MATLAB永久搜索，那么直接单击Close按钮，再在弹出的对话框中作否定回答即可。

2.用命令设置搜索路径

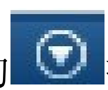
MATLAB能够将某一路径设置成可搜索路径的命令有两个：一个是 path；另一个是addpath。

针用path和addpath命令设置搜索路径的区别，下面举例说明。

>>path(path,'c:/tools/goodstuff')　　%将路径放在路径表的前面

>>addpath('c:/matlab/myfiles','-end')　%将路径放在路径表的后面

1.4.7 Workspace（工作空间）

在默认的情况下，工作空间浏览器位于MATLAB操作界面的右上方。单击Workspace窗口右上角的按钮，在出现的下拉菜单中选择Undock选项，可以将其从MATLAB主界面分离出来。另外还可以直接用鼠标将 Workspace 窗口拖离主界面，其结果如图 1-14所示。

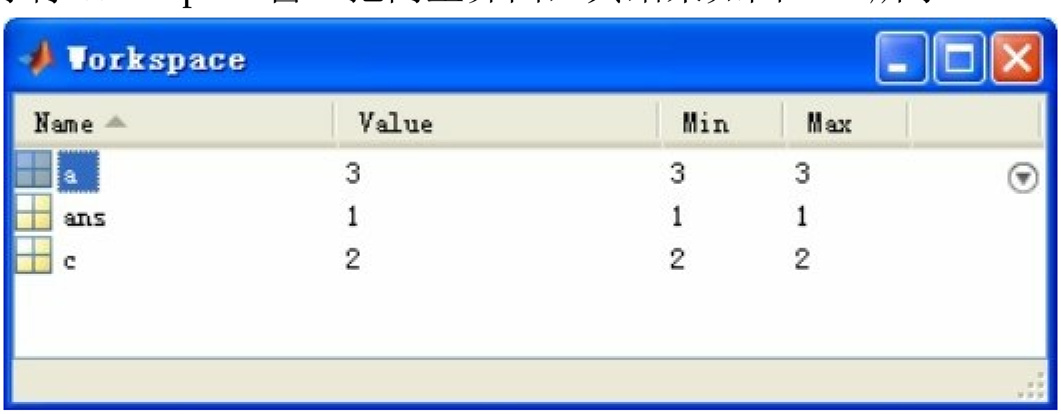


图1-14 Workspace窗口

当选中Workspace窗口中一个变量时，MATLAB界面菜单栏中的PLOTS选项就会激活，激活前后对比界面如图1-15所示。在图1-15（b）中，MATLAB强大的绘图功能有多种绘图方法供用户选择。

除了非常强大的绘图功能，Workspace 窗口还有许多其他应用功能，例如内存变量的查阅、保存和编辑等。所有这些操作都比较简单，只需要在 Workspace 窗口中选择相应的变量，然后右击鼠标，在弹出的快捷菜单中选择相应的菜单选项，如图1-16所示。



（a）未激活状态

（b）激活后状态

图1-15 PLOTS选项激活前后对比图

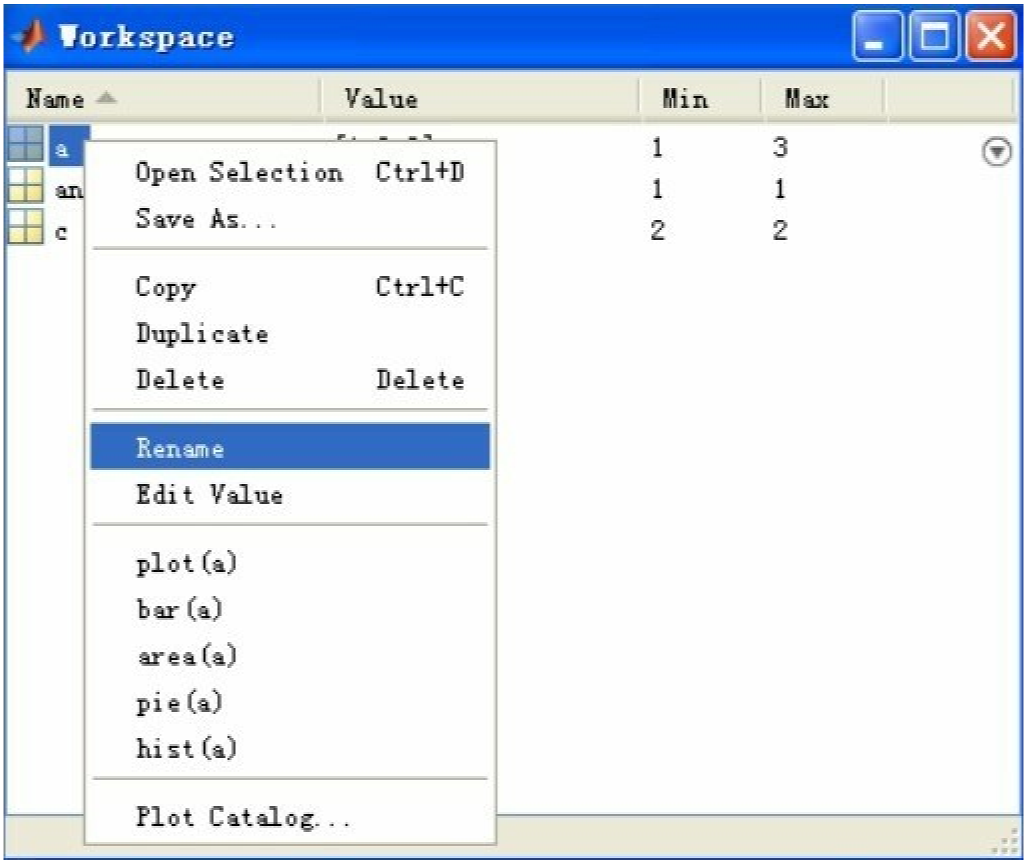


图1-16 修改变量名称

在MATLAB中，数组和矩阵都是十分重要的基础变量，因此MATLAB专门提供变量编辑器这个工具来编辑数据。

双击Workspace窗口中任意一个数组，就可以打开该数组的Variables窗口（变量编辑器），如图1-17所示。

用户可以在Variables窗口中直接编辑该变量。对于大型数组，使用变量编辑器会给用户带来很大的便利。

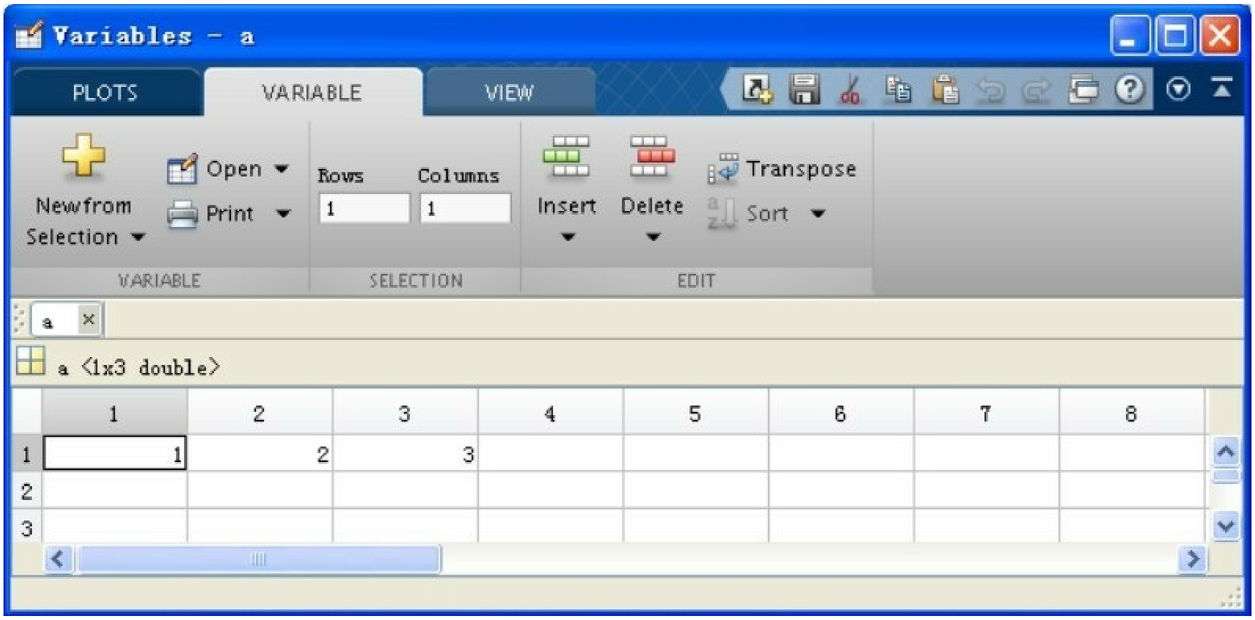


图1-17 变量编辑器窗口

1.4.8 变量的编辑命令

在 MATLAB 中，用户除了可以在工作空间浏览器中编辑内存变量之外，还可以在MATLAB的命令窗口输入相应的命令，查阅和删除内存中的变量。下面用简单的案例，说明如何在命令窗口中对变量进行操作。

【例1-1】在MATLAB命令窗口中查阅内存变量。

解：具体步骤如下。

在命令窗口中输入who和whos命令，查看内存变量的信息，如图1-18所示。

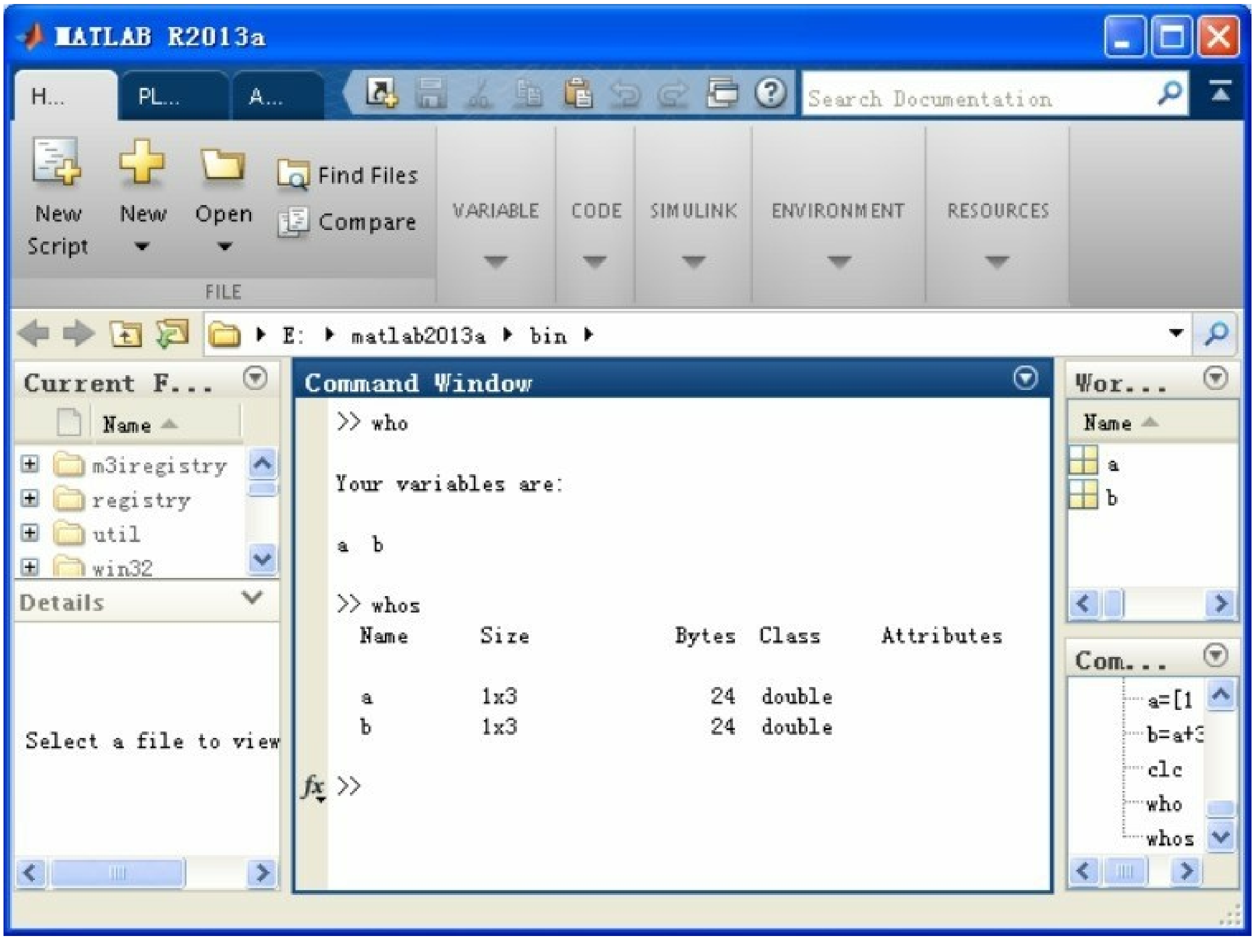


图1-18 查阅内存变量的信息

注意：who和whos命令适用于MATLAB各种版本，两个命令的区别只在于内存变量信息的详细程度。两个命令结果的列表次序随具体情况而不同。

【例1-2】在例1-1之后，在MATLAB命令窗口中删除内存变量b。

在命令窗口中输入下面命令行。

```
>> clear b
>> who
```

得到的结果如图1-19所示。

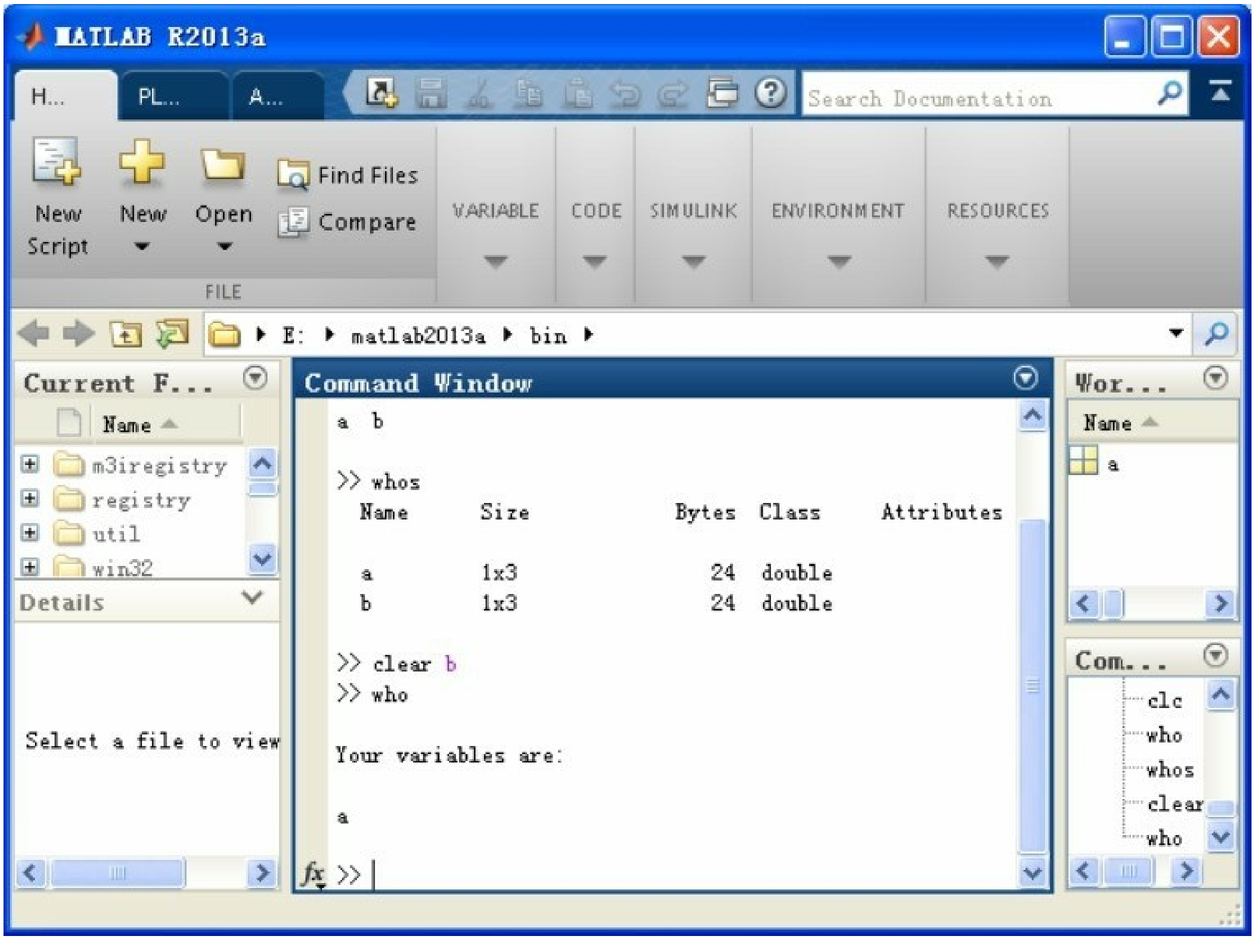


图1-19 删除内存变量

和前面的例子相比，用户可以看出，当用户运行clear命令后，将b变量从工作空间删除，而且在工作空间浏览器中也将该变量删除。

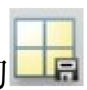
1.4.9 存取数据文件

MATLAB提供了save和load命令来实现数据文件的存取。表1-8列出了命令的常见用法。

表1-8 MATLAB文件存取的命令

命　　令	功　　能
Save Filename	将工作空间中的所有变量保存到名为 Filename 的 MAT 文件中
Save Filename x y z	将工作空间中的 x,y,z 变量保存到名为 Filename 的 MAT 文件中
Save Filename –regecp pat1 pat2	将工作空间中符合表达式要求的变量保存到名为 Filename 的 MAT 文件中
Load Filename	将名为 Filename 的 MAT 文件中的所有变量读入内存
Load Filename x y z	将名为 Filename 的 MAT 文件中的 x,y,z 变量读入内存
Load Filename –regecp pat1 pat2	将名为 Filename 的 MAT 文件中符合表达式要求的变量读入内存
Load Filename x y z -ASCII	将名为 Filename 的 ASCII 文件中的 x,y,z 变量读入内存

上表中列出了几个常见的文件存取命令，用户可以根据需要选择相应的存取命令；对于一些较少见的存取命令，用户可以查阅MATLAB帮助文件。

在MATLAB中，除了可以在命令窗口中输入相应的命令之外，也可以在工作空间中选择相应的按钮，实现数据文件的存取工作。例如，用户可以选择工具栏中的（save workspace）命令，将所有变量保存到mat文件中，如图1-20所示。

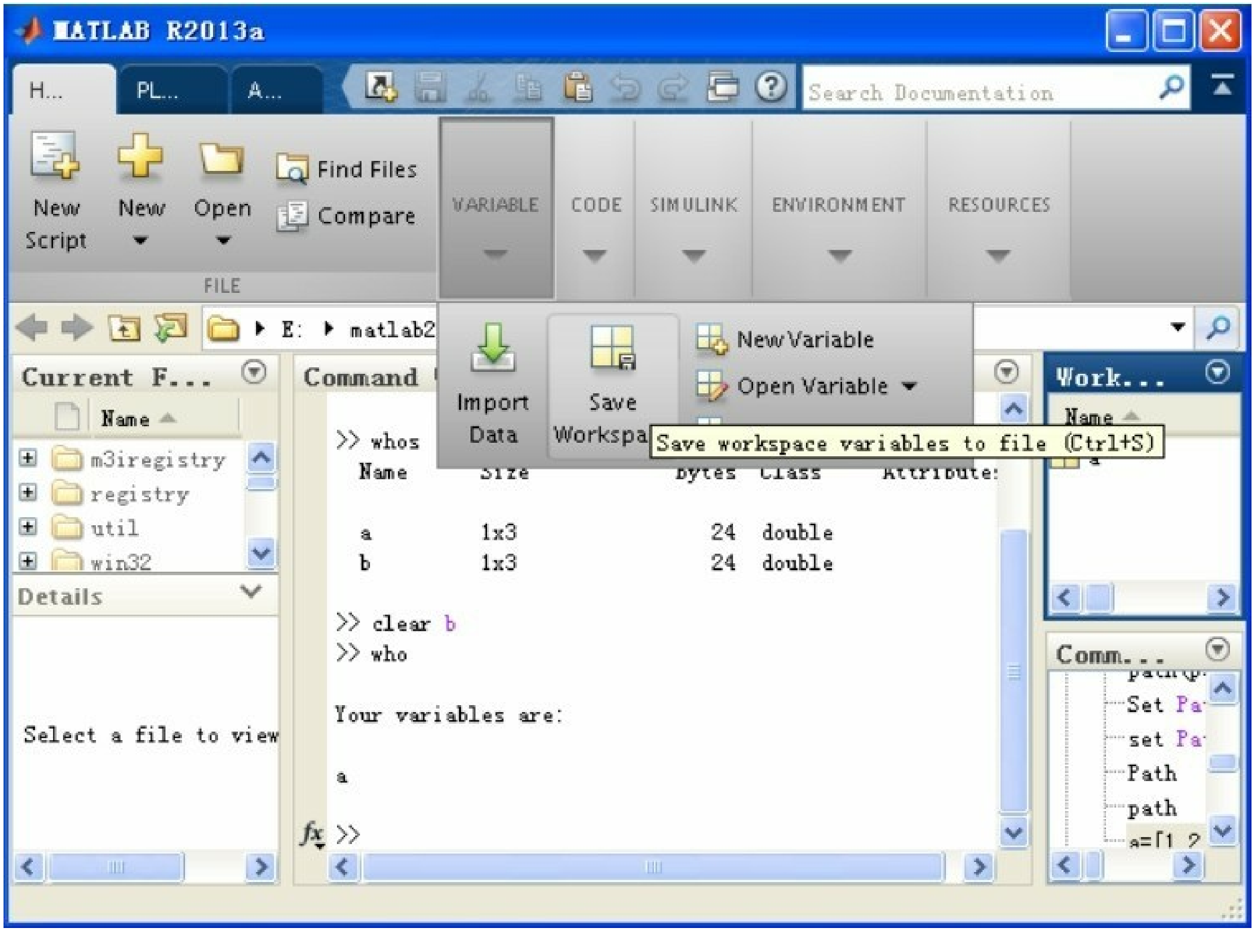

MATLAB R2013a
Search Documentation
New Script
New
Open
Find Files
Compare
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
Import Data
Save Workspa
New Variable
Open Variable
Save workspace variables to file (Ctrl+S)
Current F...
Name
m3iregistry
registry
util
win32
Details
Select a file to view
Command
>> whos
Name
a 1x3 24 double
b 1x3 24 double
>> clear b
>> who
Your variables are:
a
>>
Work...
Comm...
Set Pa
set Pa
Path
path

图1-20 保存所有变量

1.5 MATLAB R2013a的帮助系统

MATLAB的各个版本都为用户提供详细的帮助系统，可以帮助用户更好地了解和运用MATLAB。因此，不论用户是否用过 MATLAB，是否熟悉 MATLAB，都应该了解和掌握MATLAB 的帮助系统。同时，在MATLAB 6.0 版本以后，帮助系统的帮助方式和内容层次均相对之前的版本发生了本质变化，因此，用户更加有必要了解MATLAB R2013a的帮助系统。在本节中，将详细介绍MATLAB R2013a的帮助系统。

1.5.1 纯文本帮助

在MATLAB中，所有执行命令或者函数的M源文件都有较为详细的注释。这些注释都是用纯文本的形式来表示的。一般都包括函数的调用格式或者输入函数、输出结果的含义。这些帮助是最原始的（相当于最底层的源文件）。当MATLAB不同版本中函数发生变化的时候，这些文本帮助也会同步更新。

下面使用简单的例子来说明如何使用MATLAB的纯文本帮助。

【例1-3】在MATLAB中查阅帮助信息。

解：根据MATLAB的帮助体系，用户可以查阅不同范围的帮助，具体步骤如下。

（1）在MATLAB的命令窗口输入help help命令，然后按Enter键，查阅如何在MATLAB中使用help命令，如图1-21所示。

操作界面显示了如何在MATLAB中使用help命令的帮助信息，用户可以详细阅读上面的信息来解决如何使用help命令。

（2）在MATLAB的命令窗口中输入help命令，然后按Enter键，查

阅关于MATLAB系统中的所有主题的帮助信息，如图1-22所示。

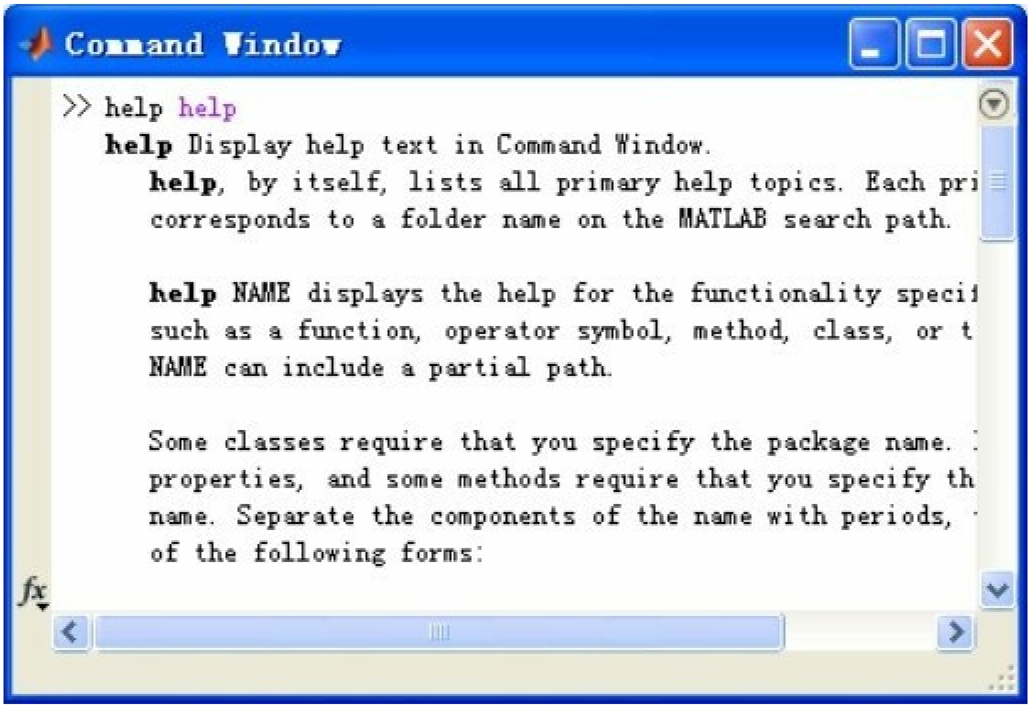


图1-21 使用help命令的帮助信息

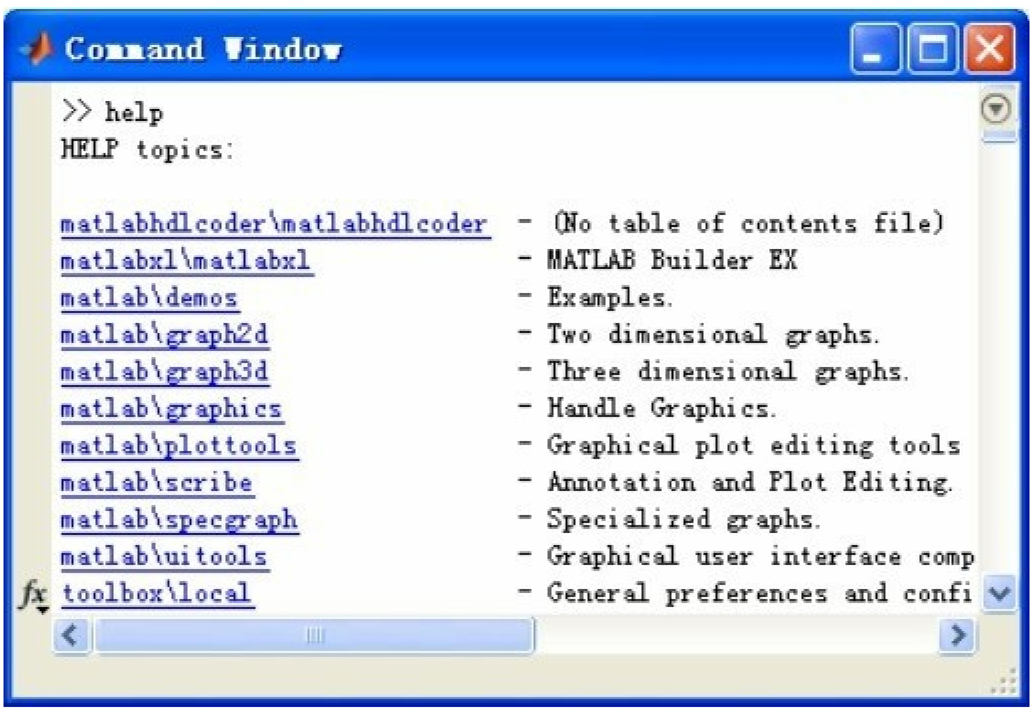


图1-22 查阅关于主题帮助信息

（3）在MATLAB 的命令窗口中输入help topic 命令，然后按Enter键，查阅关于该主题的所有帮助信息，如图1-23所示。

上面的步骤中，简单演示了如何在MATLAB中使用help命令，获得各种函数、命令的帮助信息。在实际应用中，用户可以灵活使用这些命令来搜索所需的帮助信息。

【例1-4】在 MATLAB 中搜索各命令的帮助信息，在M函数文件中搜索包含关键字jacobian的所有M函数文件名，如图1-24所示。



图1-23 查阅主题下的函数帮助信息

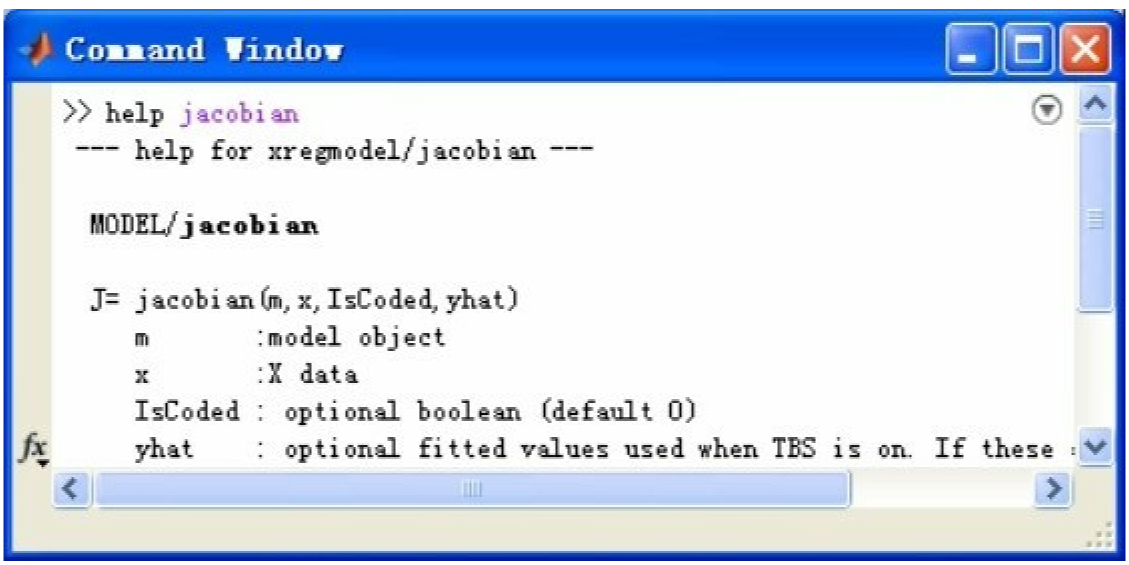


图1-24 搜索M文件的帮助文件

1.5.2 演示帮助

在MATLAB中，各个工具包都有设计好的演示程序，这组演示程

序在交互界面中运行，操作非常简便。因此，运行这组演示程序，然后研究演示程序的相关M文件，对MATLAB用户而言是十分有益的。

这种演示功能对提高用户对 MATLAB 的应用能力有着重要作用。特别对于初学者而言，不需要了解复杂的程序就可以直观地查看程序结果，可以加强用户对MATLAB的掌握能力。

在 MATLAB 的命令窗口中输入“demo”命令，就可以调用关于演示程序的帮助对话框，如图1-25所示。

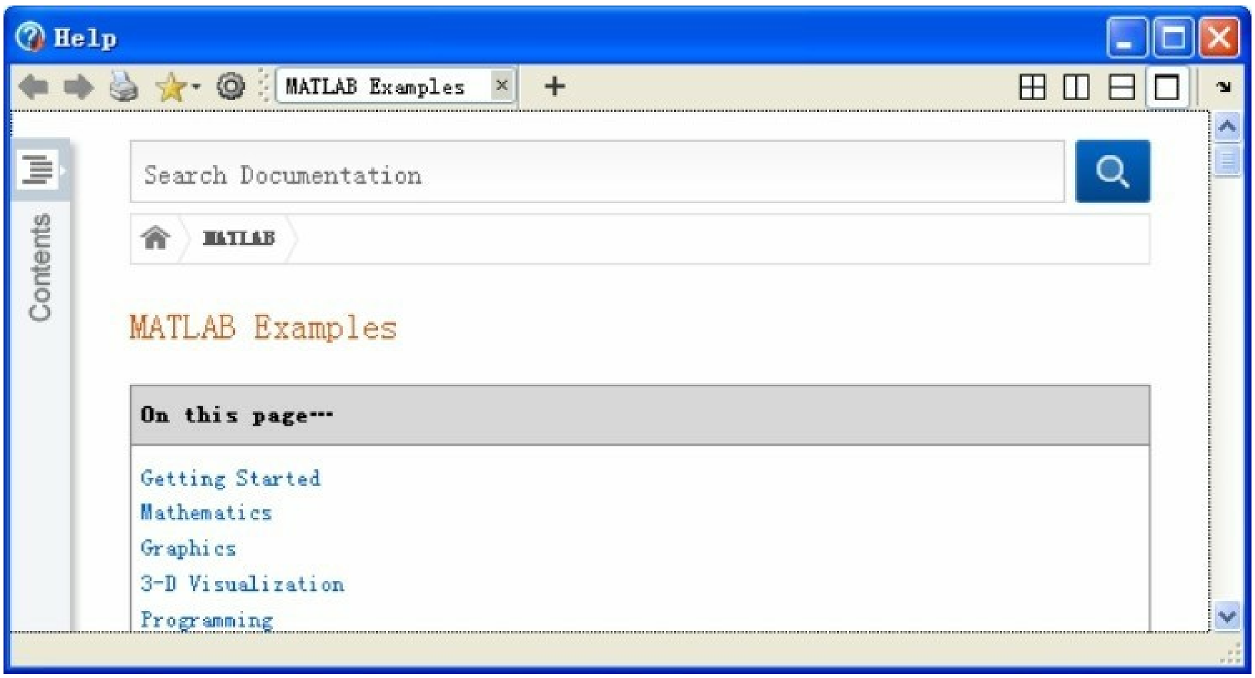


图1-25 MATLAB中的demo帮助

在上面的对话框中，单击可以显示出“demo帮助”的目录，如图1-26所示。用户可以在目录的左侧选择需要帮助的内容，也可以在图1-25中“On this page”选项中选择所需要的内容。

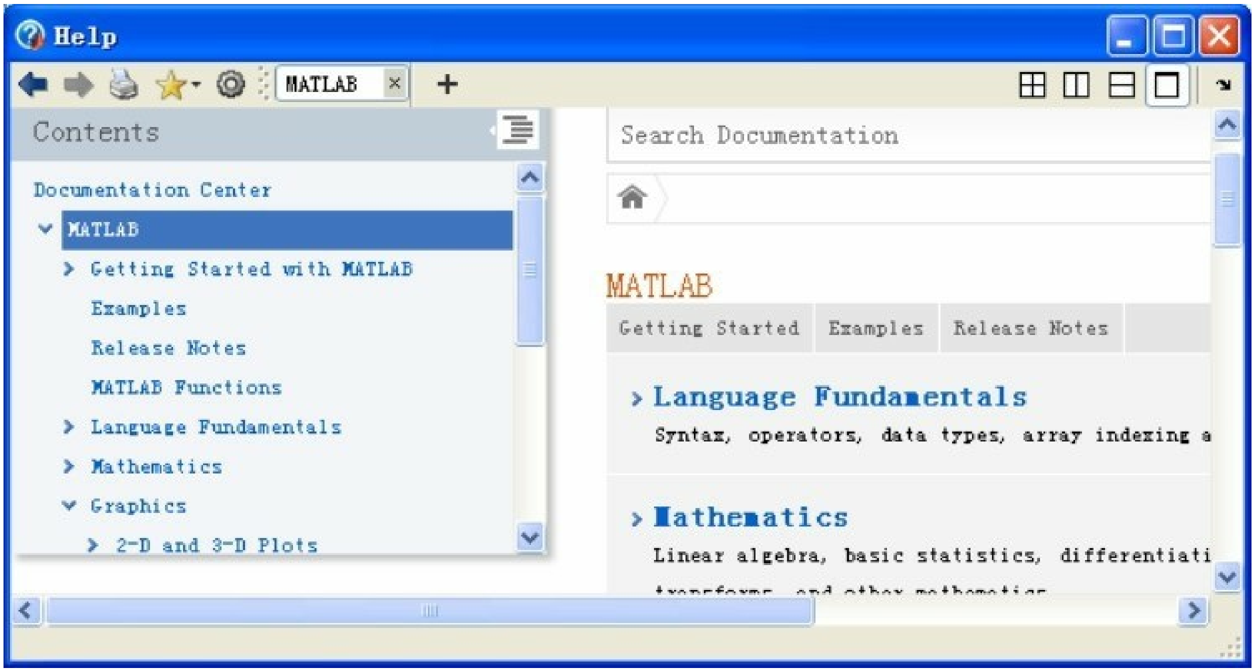


图1-26 展开demo帮助目录

例如，在“On this page”选项中选择“Graphics”选项，帮助界面会显示到“Graphics”选项处，如图1-27所示。

单击图1-27中的“Line Plotting”选项，MATLAB 会打开Line Plotting对话框，该对话框用于演示demo的交互界面，如图1-28所示。

演示该 demo 的是一个交互界面，用户可以调整该界面中图形的显示参数，这些修改的参数会出现在“Command Window”选项中，例如将Line Type 由默认的实线变为虚线，同时将Line Width 修改为3，修改后的结果如图1-29 所示。

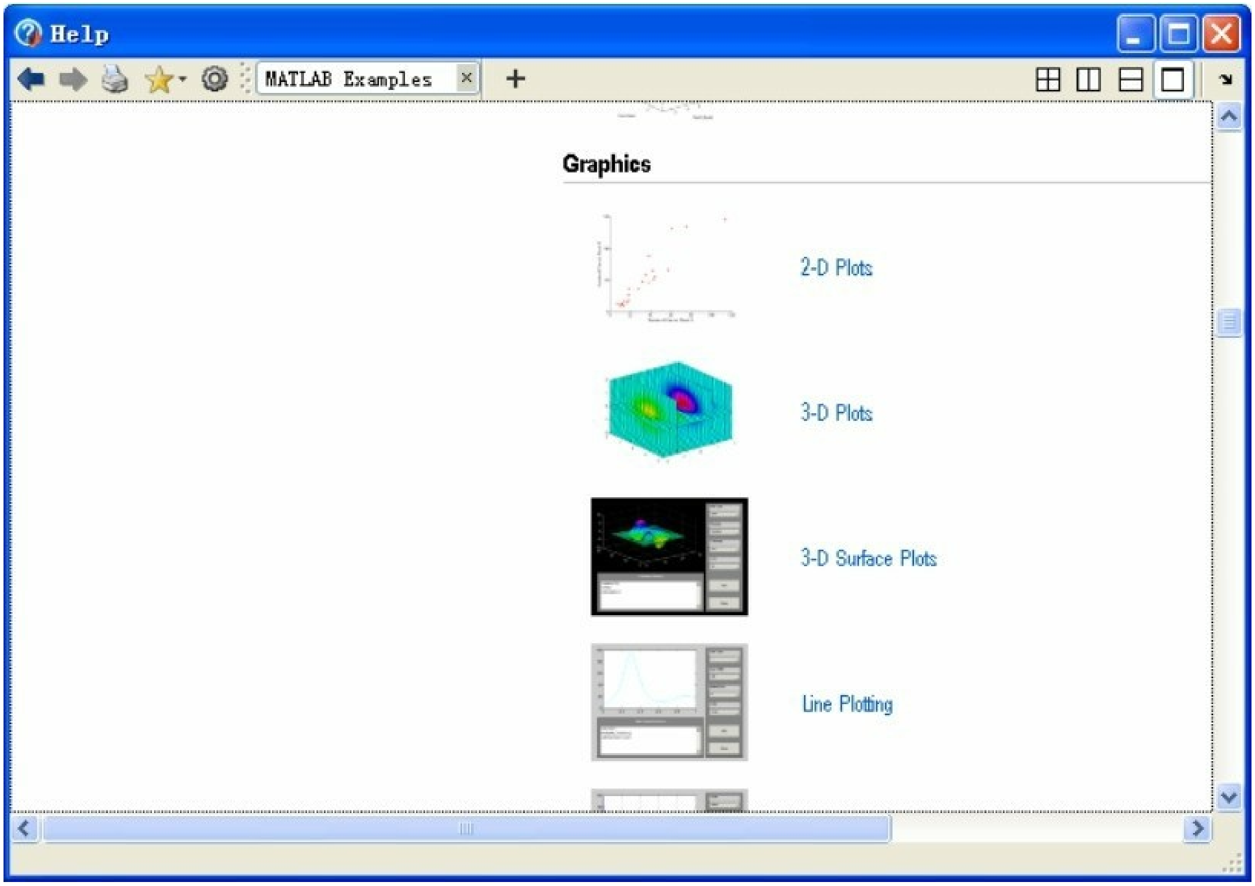


图1-27 “Graphics”选项

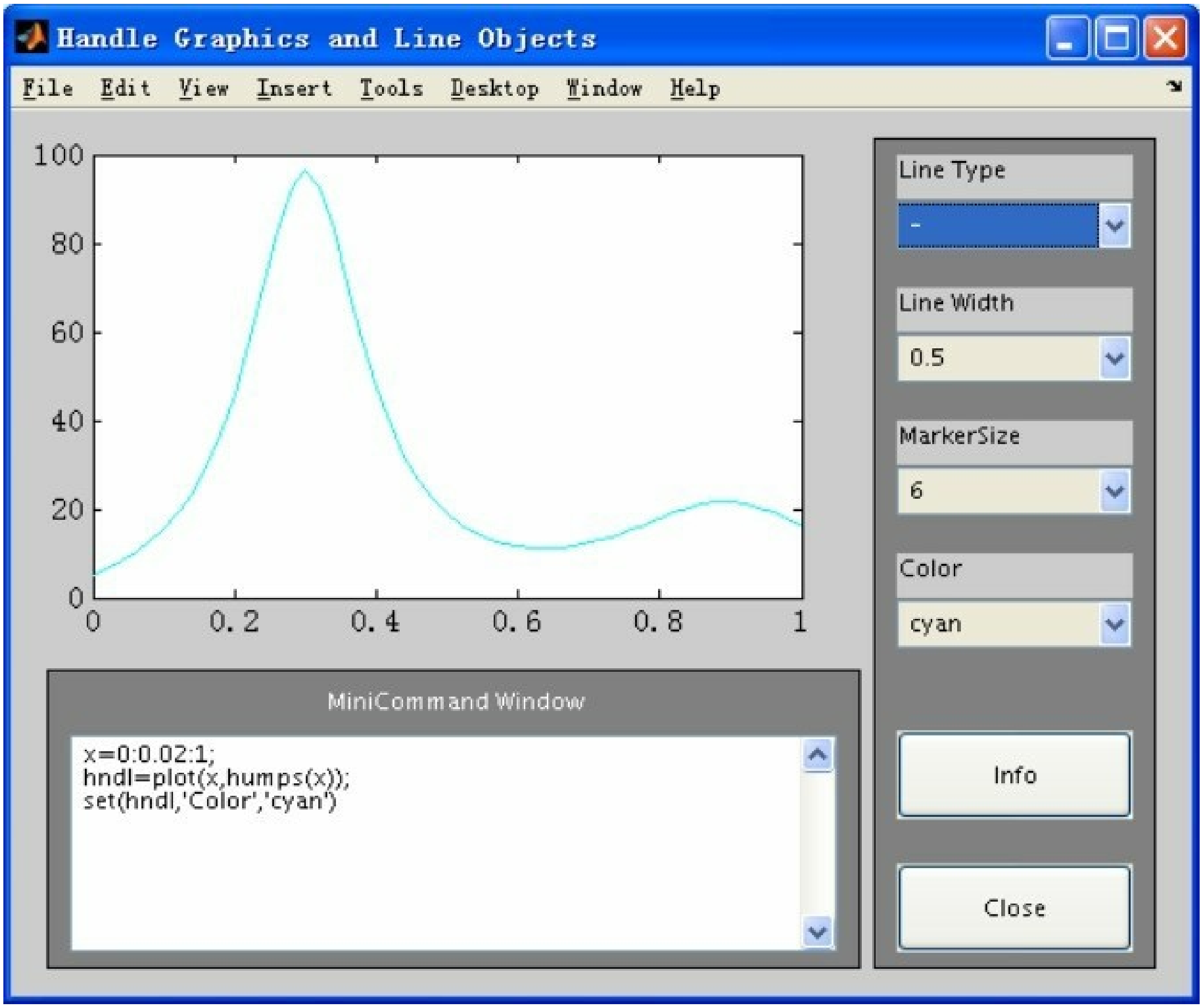


图1-28 Line Plotting对话框

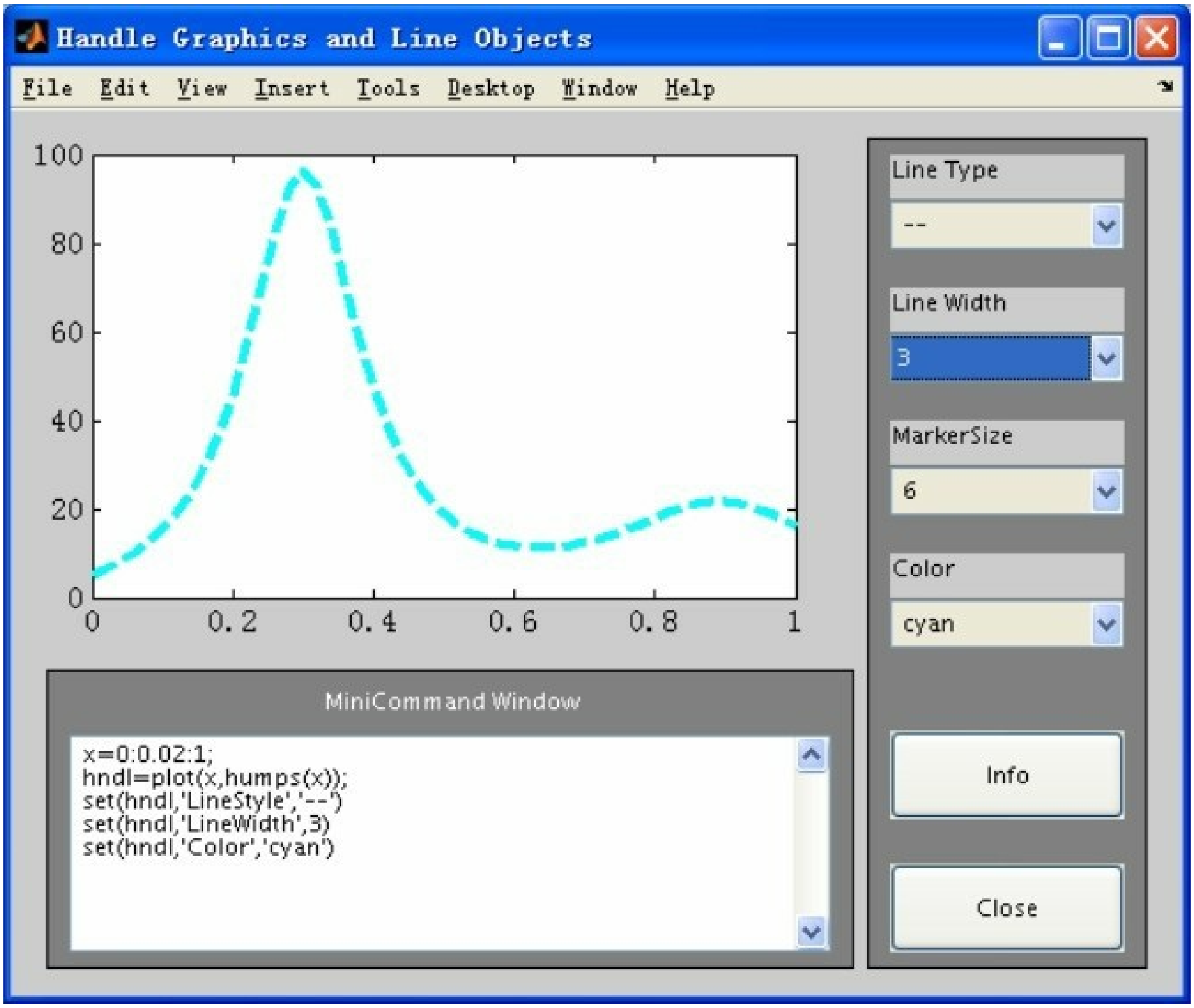


图1-29 调整曲线参数后的结果

选择图1-29中的Info 选项，可以得到Line Plotting 示例的帮助文档，如图1-30 所示。

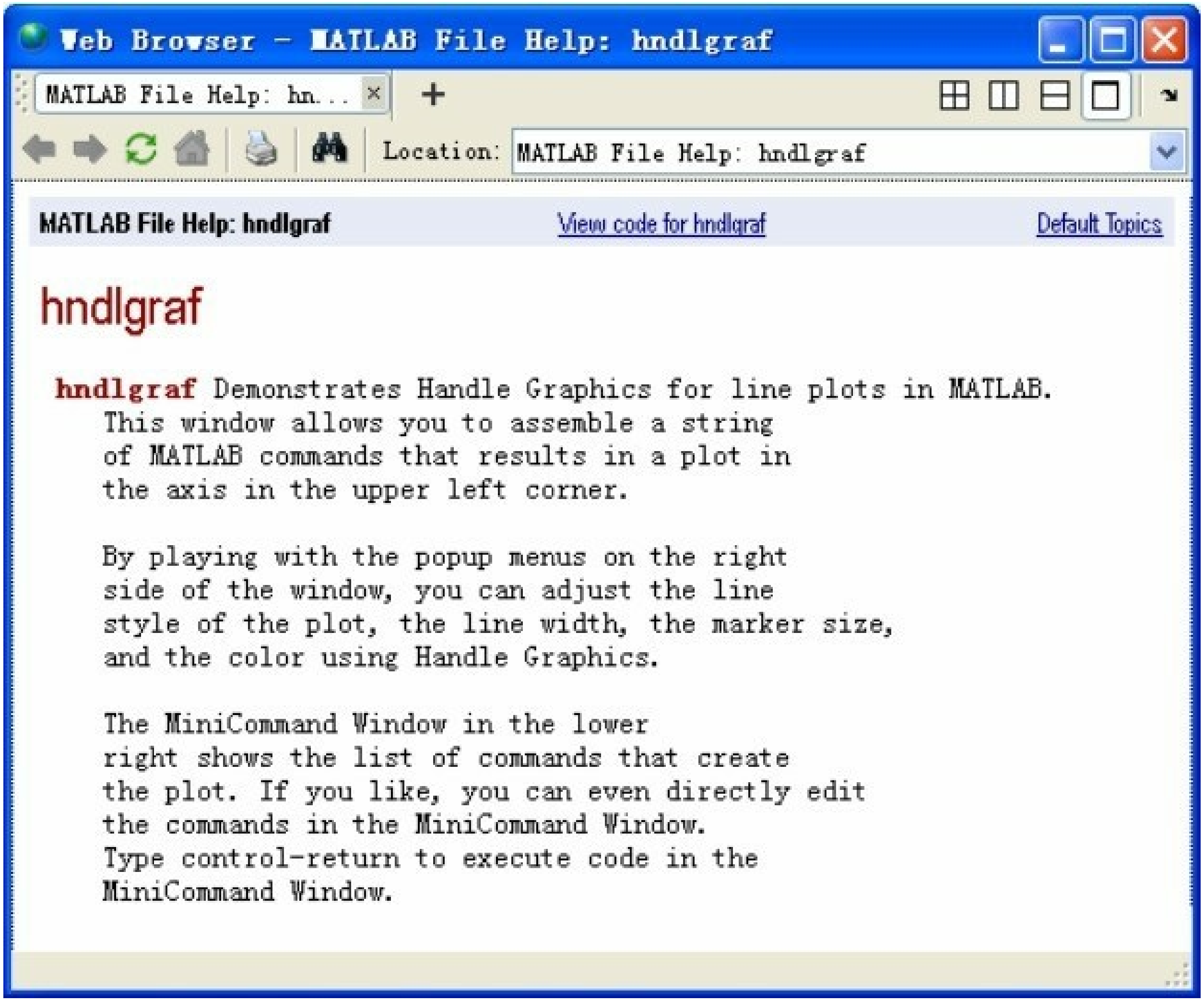


图1-30 Line Plotting示例的帮助文档

用户除了可以在打开的动态界面中演示demo之外，还可以查看该demo的程序代码。在MATLAB的Command Window窗口中输入“open hndlgraf.m”，可以查看Line Plotting示例GUI界面的MATLAB程序代码，如图1-31。

除此之外，用户也可以在图1-32所示的MATLAB工具栏中，选择Help下拉菜单中的Examples选项打开MATLAB中的demo帮助界面。

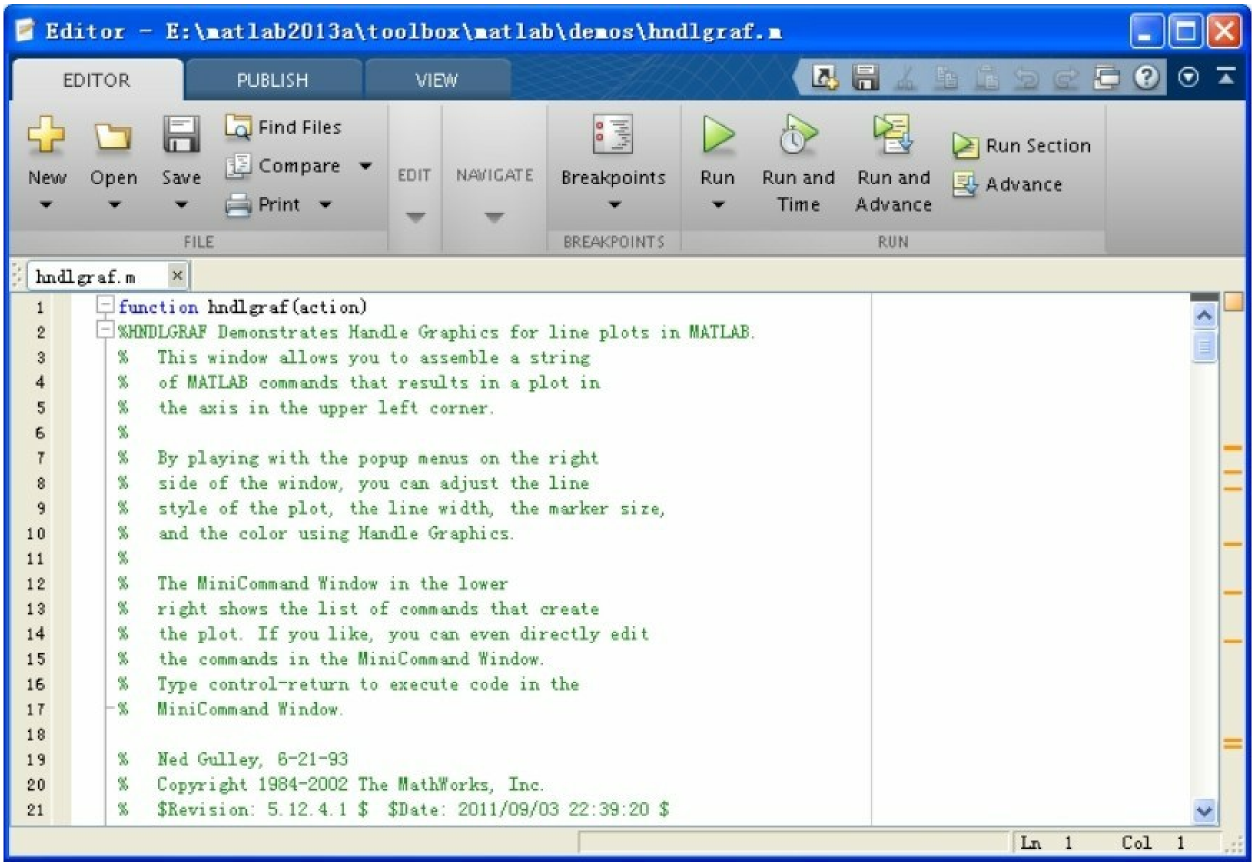


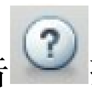
图1-31 查看demo 的程序代码



图1-32 Help下拉菜单

1.5.3 帮助导航

在 MATLAB 中提供帮助信息的“帮助导航”交互界面是 MATLAB 6.0 以后的版本的重要改进，这个帮助文件和M文件中的纯文本帮助无关，而是MATLAB专门设置的独立帮助系统。该系统对MATLAB的功能叙述的全面、系统，而且界面友好，使用方便，是用户查找帮助的重要途径。

用户可以在MATLAB的命令窗口中输入命令“helpbrowser”或者“helpdesk”，或者在操作界面中单击按钮，打开“帮助导航”交互界面，如图1-33所示。

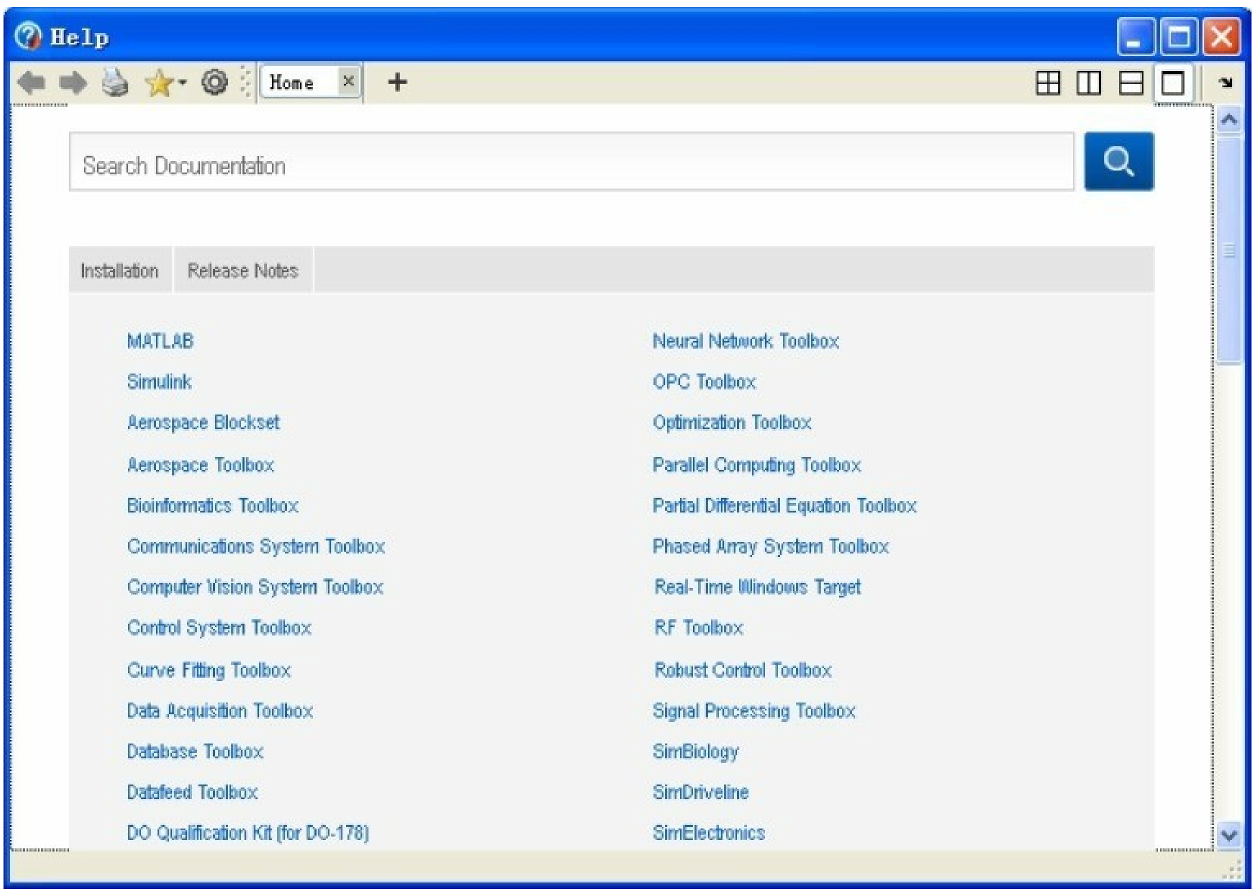


图1-33 “帮助导航/浏览器”交互界面

1.5.4 帮助文件目录窗

在默认情况下，当用户在 MATLAB 中打开“帮助导航”交互界面时，界面就会选择“Contents”选项卡。这个窗口中使用节点可展开的目录树来列出各种帮助信息，直接使用鼠标单击相应的目录条，就可以在浏览器中显示相应的标题的HTML帮助文件。

这个窗口是向用户提供全方位系统帮助的向导，层次清晰、功能划分规范，用户可以查找相应的帮助信息。例如，初学用户希望了解MATLAB，可以选择对话框中的“MATLAB”→“Getting Started with MATLAB”→“Tutorials”→“Array Indexing”选项，在浏览器中查看关于Array Indexing的帮助文件，如图1-34所示。

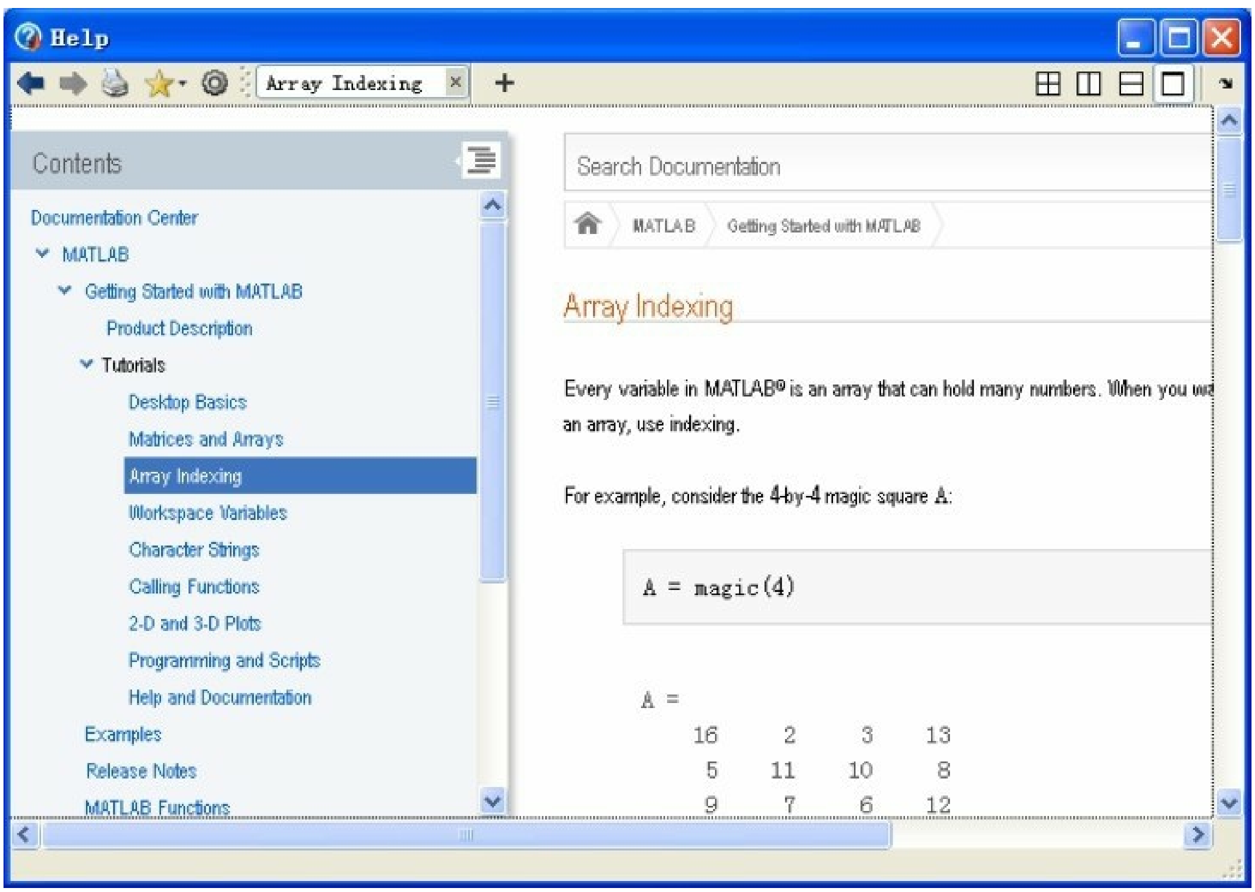


图1-34 帮助文件目录

1.5.5 帮助文件索引窗

在MATLAB中，为了提高用户使用帮助文件的效率，专门为命令、函数和一些专用术语提供索引表。用户可以在交互界面中的搜索选项中输入需要查找的名称，在其下面就会出现与此匹配的词汇列表。

同时，在浏览器的界面显示相应的介绍内容。例如，在搜索选框中输入“cos”进行搜索，得到的结果如图1-35所示。

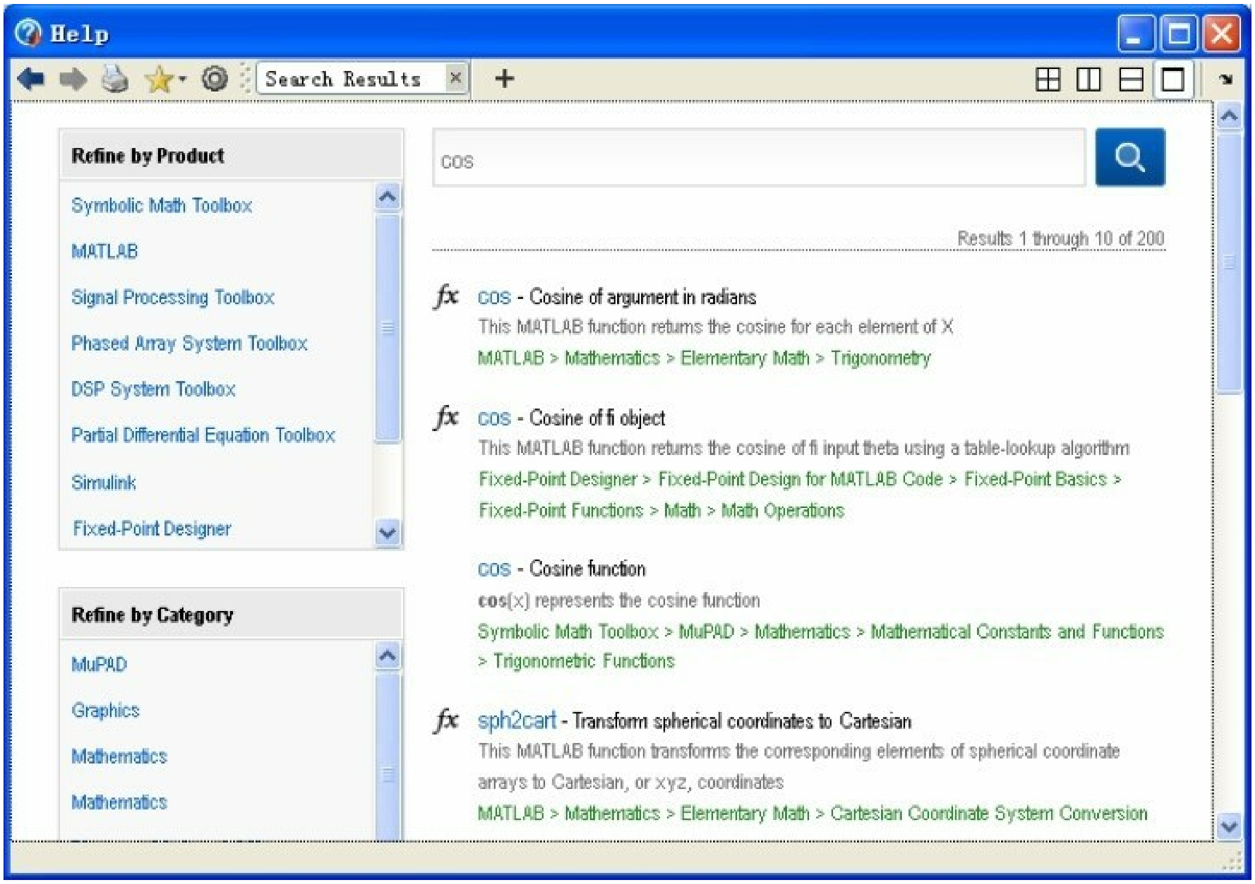


图1-35 查看cos信息

1.6 本章小结

MATLAB是一个功能多样的、高度集成的、适合科学和工程计算的软件，同时它又是一种高级程序设计语言。MATLAB的主界面集成了命令窗口、历史命令窗口、当前目录窗口和工作空间窗口等。它们既可单独使用，又可相互配合，为用户提供了十分灵活方便的操作环境。

对MATLAB各窗口的某项设置操作通常都有两条途径：一条是用MATLAB相关窗口的对话框或菜单（包括快捷菜单）；另一条是在命令窗口执行某一命令。前者的优点是方便用户与MATLAB的交互，而后者主要是考虑到程序设计的需要和方便。

希望通过本章的内容，用户能够对MATLAB有一个直观的印象。在后面的章节中，将详细介绍关于MATLAB的基础知识和基础操作方法。

第2章 MATLAB基础

数据类型、常量与变量是MATLAB程序语言入门时必须引入的一些基本概念。数组是一种在高级语言中被广泛使用的构造型数据结构。在MATLAB中，为用户提供曲线拟合图形界面，用户可以在该界面上直接进行曲线拟合。

本章除了介绍以上内容外，还将简单介绍M文件中的一些概念。

学习目标：

·了解MATLAB 的数据类型、数组等基本概念

·熟练运用曲线拟合

·掌握M文件中的基本概念和简单使用

2.1 基本概念

数据类型、常量与变量是程序语言入门时必须引入的一些基本概念，MATLAB　虽是一个集多种功能于一体的集成软件，但就其语言部分而言，这些概念同样不可缺少。

本节除了引入这些概念之外，还将介绍诸如向量、矩阵、数组、运算符、函数和表达式等概念。

2.1.1 MATLAB数据类型概述

数据作为计算机处理的对象，在程序语言中可分为多种类型，MATLAB作为一种可编程的语言当然也不例外。MATLAB的主要数据类型如图2-1所示。

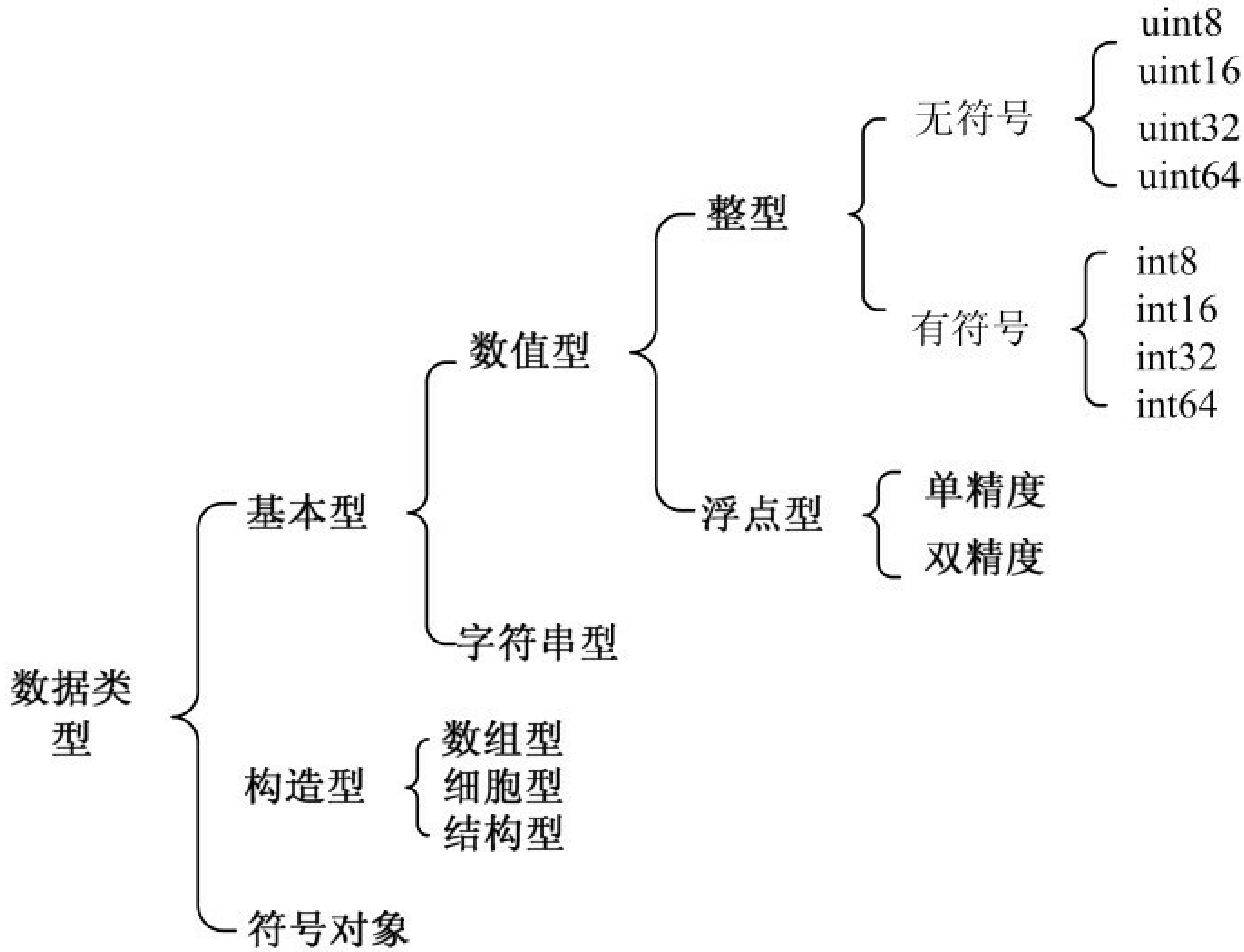


图2-1 MATLAB主要数据类型

MATLAB数值型数据划分成整型和浮点型的用意和C语言有所不同。MATLAB的整型数据主要为图像处理等特殊的应用问题提供数据类型，以便节省空间或提高运行速度。对一般数值运算，绝大多数情况是采用双精度浮点型的数据。

MATLAB的构造型数据基本上与C++的构造型数据相衔接，但它的数组却有更加广泛的含义和不同于一般语言的运算方法。

符号对象是MATLAB所特有的一类为符号运算而设置的数据类型。严格地说，它不是某一类型的数据，它可以是数组、矩阵、字符等多种形式及其组合，但它在MATLAB的工作空间中的确又是另立的一种数据类型。

在使用中，MATLAB数据类型有一个突出的特点：在对不同数据类型的变量被引用时，一般不用事先对变量的数据类型进行定义或说明，系统会依据变量被赋值的类型自动进行类型识别，这在高级语言中是极有特色的。

提示：这样处理的好处是，在书写程序时可以随时引入新的变量而不用担心会出什么问题，给应用带来了很大方便。但缺点是有失严谨，会给搜索和确定一个符号是否为变量名带来更多的时间开销。

2.1.2 常量与变量

常量是程序语句中取不变值的量，如表达式 y=0.618*x，其中就包含一个0.618 这样的数值常数，它便是一数值常量。而另一表达式s='Tomorrow and Tomorrow'中，单引号内的英文字符串“Tomorrow and Tomorrow”则是一字符串常量。

在MATLAB中，有一类常量是由系统默认给定一个符号来表示的，例如 pi，它代表圆周率π这个常数，即3.1415926…，类似于C语言中的符号常量，这些特殊常量有时又称为系统预定义的变量，见表2-1。

表2-1 MATLAB特殊常量表

常量符号	常量含义
i 或 *j*	虚数单位，定义为 $i^2=j^2=-1$
Inf 或 inf	正无穷大，由零作除数引入此常量
NaN	不定时，表示非数值量，产生于 0/0,∞/∞,0*∞等运算
pi	圆周率π的双精度表示
eps	容差变量，当某量的绝对值小于 eps 时，可以认为此量为零，即为浮点数的最小分辨率，PC 上此值为 2^{-52}
Realmin 或 realmin	最小浮点数，2^{-1022}
Realmax 或 realmax	最大浮点数，2^{1023}

变量是在程序运行中其值可以改变的量，变量由变量名来表示。在

MATLAB中变量名的命名有自己的规则，可以归纳成如下几条。

（1）变量名必须以字母开头，且只能由字母、数字或者下画线3类符号组成，不能含有空格和标点符号（如( )，。%）等。

（2）变量名区分字母的大小写。例如，“a”和“A”是不同的变量。

（3）变量名不能超过63个字符，第63 个字符后的字符被忽略，对于MATLAB 6.5 版以前的变量名不能超过31个字符。

（4）关键字（如if、while等）不能作为变量名。

（5）最好不要用表2-1中的特殊常量符号作变量名。

2.1.3 标量、向量、矩阵与数组

标量、向量、矩阵和数组是MATLAB运算中涉及的一组基本运算量。它们各自的特点及相互间的关系可以描述如下。

（1）数组不是一个数学量，而是一个用于高级语言程序设计的概念。如果数组元素按一维线性方式组织在一起，那么称其为一维数组，一维数组的数学原型是向量。

如果数组元素分行、列排成一个二维平面表格，那么称其为二维数组，二维数组的数学原型是矩阵。

如果元素在排成二维数组的基础上，再将多个行、列数分别相同的二维数组叠成一个立体表格，便形成三维数组。依此类推下去，便有了多维数组的概念。

说明：在 MATLAB 中，数组的用法与一般高级语言不同，它不借助于循环，而是直接采用运算符，有自己独立的运算符和运算法则。

（2）矩阵是一个数学概念，一般高级语言并未引入将其作为基本的运算量，但 MATLAB 是个例外。

一般高级语言是不认可将两个矩阵视为两个简单变量而直接进行加减乘除的，要完成矩阵的四则运算必须借助于循环结构。当MATLAB

将矩阵引入作为基本运算量后，运算方式并没有完全遵守上述规定。MATLAB不仅实现了矩阵的简单加减乘除运算，而且许多与矩阵相关的其他运算也因此大大简化了。

（3）向量是一个数学量，一般高级语言中也未引入，它可视为矩阵的特例。从MATLAB的工作空间窗口可以查看到：一个n维的行向量是一个1×n 阶的矩阵，而列向量则当成n×1阶的矩阵。

（4）标量的提法也是一个数学概念，但在MATLAB中，一方面可将其视为一般高级语言的简单变量来处理，另一方面又可把它当成1×1阶的矩阵，这一看法与矩阵作为MATLAB的基本运算量是一致的。

（5）在MATLAB中，二维数组和矩阵其实是数据结构形式相同的两种运算量。二维数组和矩阵的表示、建立、存储根本没有区别，区别只在它们的运算符和运算法则不同。

例如，向命令窗口中输入a=[1 2;3 4]这个量，实际上它有两种可能的角色：矩阵a或二维数组 a。这就是说，单从形式上是不能完全区分矩阵和数组的，必须再看它使用什么运算符与其他量之间进行运算。

（6）数组的维和向量的维是两个完全不同的概念。数组的维是从数组元素排列后所形成的空间结构去定义的：线性结构是一维，平面结构是二维，立体结构是三维，当然还有四维以至多维。向量的维相当于一维数组中的元素个数。

下面介绍几个向量和矩阵的代数运算示例。

【例2-1】向量点积运算。

解：在MATLAB的Command Window窗口输入以下命令。

```
>> A=1:10;
>> B=linspace(1,10,10);
>> AT=A';
>> BT=B';
>> e=dot(A,B)
```

```
>> f=dot(AT,BT)
```

其运算结果如下。

```
e =
   385
f =
   385
```

在数学描述中，向量A、B的叉积是一新向量C，C的方向垂直于A与B所决定的平面。

【例2-2】合法向量叉积运算。

解：在Command Window窗口输入以下命令。

```
>>A=1:3,
>>B=3:5
>>E=cross(A,B) %该函数计算的是A、B叉积后各分量的元素值，且A、B只能是三维向量
```

其运算结果如下。

```
A =
   1    2    3
B =
   3    4    5
E =
   -2    4        -2
```

【例2-3】求给定矩阵的行列式值。

解：在Command Window窗口输入以下命令并得到其结果如下所示。

```
>> A=[3 2 4;1 -1 5;2 -1 3],D1=det(A)
A =
   3              2              4
```

```
    1          -1          5
    2          -1          3
D1 =
    24
>> B=ones(3),D2=det(B),C=pascal(4),D3=det(C)
B =
    1     1     1
    1     1     1
    1     1     1
D2 =
    0
C =
    1     1     1     1
    1     2     3     4
    1     3     6    10
    1     4    10    20
D3 =
    1
```

【例2-4】已知矩阵$A=\begin{pmatrix}1 & 3\\ 2 & -1\end{pmatrix}, B=\begin{pmatrix}3 & 0\\ 1 & 2\end{pmatrix}$，求A+B，2A，2A−3B，AB。

解：在Command Window窗口输入以下命令。

```
>> A=[1 3;2 -1];
>>B=[3 0;1 2];
```

得到结果如下所示。

```
>> A+B
ans =
    4         3
    3         1
>> 2*A
ans =
    2         6
    4           -2
```

因为矩阵加减运算的规则是对应元素相加减，所以参与加减运算的矩阵必须是同阶矩阵。数与矩阵的加减乘除的规则很简单，但矩阵相乘有定义的前提是两矩阵内阶相等。

2.1.4 运算符

MATLAB运算符可分为3大类，即算术运算符、关系运算符和逻辑运算符。下面分类给出它们的运算符和运算法则。

1.算术运算符

算术运算因所处理的对象不同，分为矩阵和数组算术运算两类。表2-2 给出的是矩阵算术运算的符号、名称、示例和使用说明，表 2-3 给出的是数组算术运算的符号、名称、示例和使用说明。

表2-2 矩阵算术运算符

运算符	名称	示例	法则或使用说明
+	加	$C=A+B$	矩阵加法法则，即 $C(i,j)=A(i,j)+B(i,j)$
–	减	$C=A-B$	矩阵减法法则，即 $C(i,j)=A(i,j)-B(i,j)$
*	乘	$C=A*B$	矩阵乘法法则
/	右除	$C=A/B$	定义为线性方程组 $X*B=A$ 的解，即 $C=A/B=A*B^{-1}$
\	左除	$C=A\backslash B$	定义为线性方程组 $A*X=B$ 的解，即 $C=A\backslash B=A^{-1}*B$
^	乘幂	$C=A\text{^}B$	A、B 其中一个为标量时有定义
'	共轭转置	$B=A'$	B 是 A 的共轭转置矩阵

表2-3 数组算术运算符

运 算 符	名 称	示 例	法则或使用说明
.*	数组乘	$C=A.*B$	$C(i,j)=A(i,j)*B(i,j)$
./	数组右除	$C=A./B$	$C(i,j)=A(i,j)/B(i,j)$
.\	数组左除	$C=A.\backslash B$	$C(i,j)=B(i,j)/A(i,j)$
.^	数组乘幂	$C=A.\hat{}B$	$C(i,j)=A(i,j)\hat{}B(i,j)$
.'	转置	$A.'$	将数组的行摆放成列，复数元素不做共轭

针对表2-2和表2-3需要说明以下几点。

（1）矩阵的加、减、乘运算是严格按矩阵运算法则定义的，而矩阵的除法虽和矩阵求逆有关系，但却分了左、右除，因此不是完全等价的。乘幂运算更是将标量幂扩展到矩阵可作为幂指数。总的来说，MATLAB接受了线性代数已有的矩阵运算规则，但又不仅止于此。

（2）表 2-3 中并未定义数组的加减法，是因为矩阵的加减法与数组的加减法相同，所以未做重复定义。

（3）不论是加减乘除，还是乘幂，数组的运算都是元素间的运算，即对应下标元素一对一的运算。

（4）多维数组的运算法则，可依元素按下标一一对应参与运算的原则将表2-3推广。

2.关系运算符

MATLAB关系运算符列在表2-4中。

表2-4 关系运算符

运算符	名 称	示 例	法则或使用说明
<	小于	$A<B$	1．A、B 都是标量，结果是或为 1(真)或为 0(假)的标量 2．A、B 若一个为标量，另一个为数组，标量将与数组各元素逐一比较，结果为与运算数组行列相同的数组，其中各元素取值或 1 或 0 3．A、B 均为数组时，必须行、列数分别相同，A 与 B 各对应元素相比较，结果为与 A 或 B 行列相同的数组，其中各元素取值或 1 或 0 4．==和～=运算对参与比较的量同时比较实部和虚部，其他运算只比较实部
<=	小于等于	$A<=B$	
>	大于	$A>B$	
>=	大于等于	$A>=B$	
==	恒等于	$A==B$	
～=	不等于	$A\sim=B$	

需要明确指出的是，MATLAB的关系运算虽可看成矩阵的关系运

算，但严格地讲，把关系运算定义在数组基础之上更为合理。因为从表2-4　所列法则不难发现，关系运算是元素一对一的运算结果。数组的关系运算向下可兼容一般高级语言中所定义的标量关系运算。

3.逻辑运算符

逻辑运算在MATLAB中同样需要，为此MATLAB定义了自己的逻辑运算符，并设定了相应的逻辑运算法则，见表2-5。

表2-5 逻辑运算符

运算符	名 称	示 例	法则或使用说明
&	与	*A*&*B*	1．*A*、*B* 都为标量，结果是或为 1(真)或为 0(假)的标量 2．*A*、*B* 若一个为标量，另一个为数组，标量将与数组各元素逐一做逻辑运算，结果为与运算数组行列相同的数组，其中各元素取值或 1 或 0 3．*A*、*B* 均为数组时，必须行、列数分别相同，*A* 与 *B* 各对应元素做逻辑运算，结果为与 *A* 或 *B* 行列相同的数组，其中各元素取值或 1 或 0 4．先决与、先决或是只针对标量的运算
|	或	*A*|*B*	
～	非	～*A*	
&&	先决与	*A*&&*B*	
||	先决或	*A*||*B*	

同样地，MATLAB的逻辑运算也是定义在数组的基础之上，向下可兼容一般高级语言中所定义的标量逻辑运算。为提高运算速度，MATLAB还定义了针对标量的先决与和先决或运算。

先决与运算是当该运算符的左边为1（真）时，才继续与该符号右边的量做逻辑运算。先决或运算是当运算符的左边为1（真）时，就不需要继续与该符号右边的量做逻辑运算，而立即得出该逻辑运算结果为1（真）；否则，就要继续与该符号右边的量运算。

4.运算符的优先级

和其他高级语言一样，当用多个运算符和运算量写出一个MATLAB表达式时，运算符的优先次序是一个必须明确的问题。表2-6列出了运算符的优先次序。

表2-6 MATLAB运算符的优先次序

优先次序	运 算 符
最高	'（转置共轭）、^（矩阵乘幂）、.'（转置）、.^ （数组乘幂）
↓	～（逻辑非)
	* 、/（右除）、\（左除）、.* （数组乘）、./ （数组右除）、.\ （数组左除）
	＋、－、:（冒号运算）
	<、<=、>、>=、==（恒等于）、～=（不等于）
	&（逻辑与）
	|（逻辑或）
	&&（先决与）
最低	||（先决或）

MATLAB运算符的优先次序在表2-6中依从上到下的顺序，分别由高到低。而表中同一行的各运算符具有相同的优先级，而在同一级别中又遵循有括号先括号运算的原则。

2.1.5 命令、函数、表达式和语句

有了常量、变量、数组和矩阵，再加上各种运算符即可编写出多种MATLAB的表达式和语句。但在MATLAB的表达式或语句中，还有一类对象会时常出现，那便是命令和函数。

1.命令

命令通常就是一个动词，在第1章中已经有过接触，例如clear命令，用于清除工作空间。还有的可能在动词后带有参数，例如“addpath F:\MATLAB 文件\M文件-end”命令，用于添加新的搜索路径。在MATLAB 中，命令与函数都组织在函数库里，函数库 general就是专门用来存放通用命令的。一个命令也是一条语句。

2.函数

函数对MATLAB而言，有相当特殊的意义，这不仅因为函数在MATLAB中应用面广，更在于其种类和数量多。仅就MATLAB的基本部分而言，其所包括的函数类别就达二十多个，而每一类中又有少则几

个，多则几十个函数。

除了基本部分之外，还有各种工具箱，而工具箱实际上也是由一组组用于解决专门问题的函数构成。不包括MATLAB网站上外挂的工具箱函数，就目前MATLAB自带的工具箱已多达几十种，可见MATLAB其函数之多。从某种意义上说，函数就代表了MATLAB， MATLAB全靠函数来解决问题。

函数最一般的引用格式如下。

函数名（参数1，参数2，...）

例如，引用正弦函数就书写成sin(A)，A就是一个参数，它可以是一个标量，也可以是一个数组，而对数组求其正弦是针对其中各元素求正弦，这是由数组的特征决定的。

3.表达式

用多种运算符将常量、变量（含标量、向量、矩阵和数组等）、函数等多种运算对象连接起来构成的运算式就是MATLAB的表达式。

例如，A+B&C−sin(A·pi)就是一个表达式。请分析它与表达式(A+B)&C−sin(A·pi)有无区别。

4.语句

在MATLAB中，表达式本身即可视为一个语句。而典型的MATLAB语句是赋值语句，其一般的结构如下。

变量名=表达式

例如，F=(A+B)&C−sin(A·pi)就是一个赋值语句。

除赋值语句外，MATLAB还有函数调用语句、循环控制语句、条件分支语句等。这些语句将会在后面章节逐步介绍。

2.2 MATLAB中的数组

MATLAB中数组可以说无处不在，任何变量在MATLAB中都是以数组形式存储和运算的。按照数组元素个数和排列方式，MATLAB中的数组可以分为以下几种。

·没有元素的空数组（empty array）。

·只有一个元素的标量（scalar），它实际上是一行一列的数组。

·只有一行或者一列元素的向量（vecor），分别叫做行向量和列向量，也统称为一维数组。

·普通的具有多行多列元素和二维数组。

·超过二维的多维数组（具有行、列、页等多个维度）。

按照数组的存储方式，MATLAB中的数组可以分为：普通数组和稀疏数组（常称为稀疏矩阵）。稀疏矩阵适用于那些大部分元素为0，只有少部分非零元素的数组的存储。主要是为了提高数据存储和运算的效率。

2.2.1 数组的保存和装载

许多实际应用中的数组都是很庞大的，而且当操作步骤较多，不能在短期内完成，需要多次分时进行时，这些庞大的数组的保存和装载就是一个重要问题了，因为每次在进行操作前对数组进行声明和赋值，需要很庞大的输入工作量，一个好的解决方法是将数组保存在文件中，每次需要时进行装载。

MATLAB中提供了内置的把变量保存在文件中的方法，最简单易用的是将数组变量保存为二进制的.mat文件，用户可以通过save命令将

工作区中指定的变量存储在.mat文件中。

（1）save命令的一般语法如下。

save <filename><var1><var2>…<varN>

其作用是把var1　var2…varN指定的工作区变量存储在filename指定名称的.mat文件中。

通过save存储到.mat文件中的数组变量，在使用前可以用load命令装载到工作区。

（2）load命令的一般语法如下。

load <filename><var1><var2>…<varN>

其作用是把当前目录下存储在filename.mat　文件中的var1　var2…varN指定的变量装载到MATLAB工作区中。

2.2.2 数组索引和寻址

数组操作中最频繁遇到的就是对数组的某个具体位置上的元素进行访问和重新赋值。这涉及定位数组中元素的位置，也就是数组索引和寻址的问题。

MATLAB中数组元素的索引方式包括数字索引和逻辑索引两类。

1.数字索引方式

MATLAB中，普通二维数组元素的数字索引方式又可以分为两种：双下标（也叫全下标）索引和单下标索引。

双下标索引方式，顾名思义，就是用两个数字（自然数）来定位元素的位置。实际上就是用一个有序数对来表征元素位置，第一个数字指定元素所在的行位置，第二个数字指定元素所在的列。两个表示元素位置的索引数字之间用逗号分隔，并用圆括号括起来，紧跟在数组变量名后，就可以访问此数字索引指定的位置上的数组元素了。

例如，对于3行2列的数组A，A(3,1)表示数组A的第3行第1列的元

素，A(1,2)表示数组A的第1行第2列的元素。

相应地，单下标索引方式就是用一个数字来定位数组元素。实际上，单下标索引和双下标索引是一一对应的，对一个已知尺寸的数组，任一个单下标索引数字都可以转换成确定的双下标索引。对于m 行n 列的数组A，A(x,y)实际上对应于A((y−1)·m+x)。

例如，对于3行2列的数组A，A(3,1)用单下标索引表示就是A(3)，A(1,2)用单下标索引表示就是A(4)。

MATLAB中单下标索引方式实际上采用了列元素优先的原则，即对于m行n列的数组A，第一列的元素的单下标索引依次为A(1)，A(2)，A(3)，…，A(m)。第二列的元素的单下标索引依次为A(m+1)，A(m+2)，A(m+3)，…，A(2m)，依此类推。

这两种数字索引方式中的数字索引也可以是一个数列，从而实现访问多个数组元素的目的，这通常可以通过运用冒号或一维数组来实现。

【例2-5】数组元素的索引与寻址。

解：在Command Window窗口输入以下命令。

```
>>A=[42 56;31 7 0;1245 7823]    %创建数组
A =
   4    2    5    6
   3    1    7    0
   12    45    78    23
>>A(2,3)                    %双下标索引访问数组第2行第3列元素
ans =
   7
>>A(7)=100                  %对数组第7个元素（即第1行第3列）重新赋值
A =
   4    2    100    6
```

    3   1   7   0
    12   45   78   23

通过例题可以看到，利用下标索引的方法，用户可以访问特定位置上的数组元素的值，或者对特定位置的数组元素重新赋值。

2.单下标索引和双下标索引的转换

单下标索引和双下标索引之间，可以通过MATLAB提供的函数进行转换。

把双下标索引转换为单下标索引，需要用sub2ind命令，其语法如下。

```
IND=sub2ind(siz,I,J)
```

其中siz是一个1行2列的数组，指定转换数组的行列尺寸，一般可以用size(A)来表示：I和J分别是双下标索引中的两个数字；IND则为转换后的单下标数字。

把单下标索引转换为双下标索引，需要用ind2sub命令，其语法如下。

```
[I,J]= ind2sub(siz,IND)
```

各变量意义同上。

【例2-6】单-双下标转换。

解：在命令窗口输入以下命令。

```
>> A=rand(3,5)
A =
   0.6948    0.0344    0.7655    0.4898    0.7094
   0.3171    0.4387    0.7952    0.4456    0.7547
   0.9502    0.3816    0.1869    0.6463    0.2760
>> IND=sub2ind(size(A),2,4)
IND =
   11
```

```
>> A(IND)
ans =
   0.4456
>> [I,J]=ind2sub(size(A),13)
I =
   1
J =
   5
```

可以看到，sub2ind函数和ind2sub函数实现了单-双下标的转换，需要注意的是，ind2sub函数需要指定两个输出参数的接收变量。但由于MATLAB 中小写字母 i，j 默认是用作虚数单位，因此最好是不用小写字母i，j来接收转换后的下标数字。

3.逻辑索引方式

除了这种双下标和单下标的数字索引外，MATLAB中访问数组元素，还可以通过逻辑索引的方式，通常是通过比较关系运算产生一个满足比较关系的数组元素的索引数组（实际上是一个由0，1组成的逻辑数组），然后利用这个索引数组来访问原数组，并进行重新赋值等操作。

【例2-7】逻辑索引。

解：在命令窗口输入以下命令。

```
>>A=rand(5)          %创建数组
A =
   0.6797    0.9597    0.2551    0.5472    0.2543
   0.6551    0.3404    0.5060    0.1386    0.8143
   0.1626    0.5853    0.6991    0.1493    0.2435
   0.1190    0.2238    0.8909    0.2575    0.9293
   0.4984    0.7513    0.9593    0.8407    0.3500
>>B=A>0.8          %通过比较关系运算产生逻辑索引
```

```
B =
   0   1   0   0   0
   0   0   0   0   1
   0   0   0   0   0
   0   0   1   0   1
   0   0   1   1   0
>>A(B)=0            %通过逻辑索引访问原数组元素，并重新赋值
A =
   0.6797        0   0.2551   0.5472   0.2543
   0.6551   0.3404   0.5060   0.1386        0
   0.1626   0.5853   0.6991   0.1493   0.2435
   0.1190   0.2238        0   0.2575        0
   0.4984   0.7513        0        0   0.3500
```

2.2.3 数组的扩展和裁剪

在许多操作过程中，需要对数组进行扩展或剪裁，数组扩展是指在超出数组现有尺寸的位置添加新元素；裁剪是指从现有数据中提取部分，产生一个新的小尺寸的数组。

1.Array Editor（变量编辑器）

Array Editor 是 MATLAB 提供的对数组进行编辑的交互式图形界面工具。双击MATLAB 默认界面下工作区面板下的某一变量，都能打开Array Editor，从而进行数组元素的编辑。

Array Editor界面类似于电子表格界面，每一个单元格就是一个数组元素。当单击超出数组当前尺寸的位置的单元格，并输入数据赋值时，实际上就是在该位置添加数组元素，即进行了数组的扩展操作，如图2-2所示。

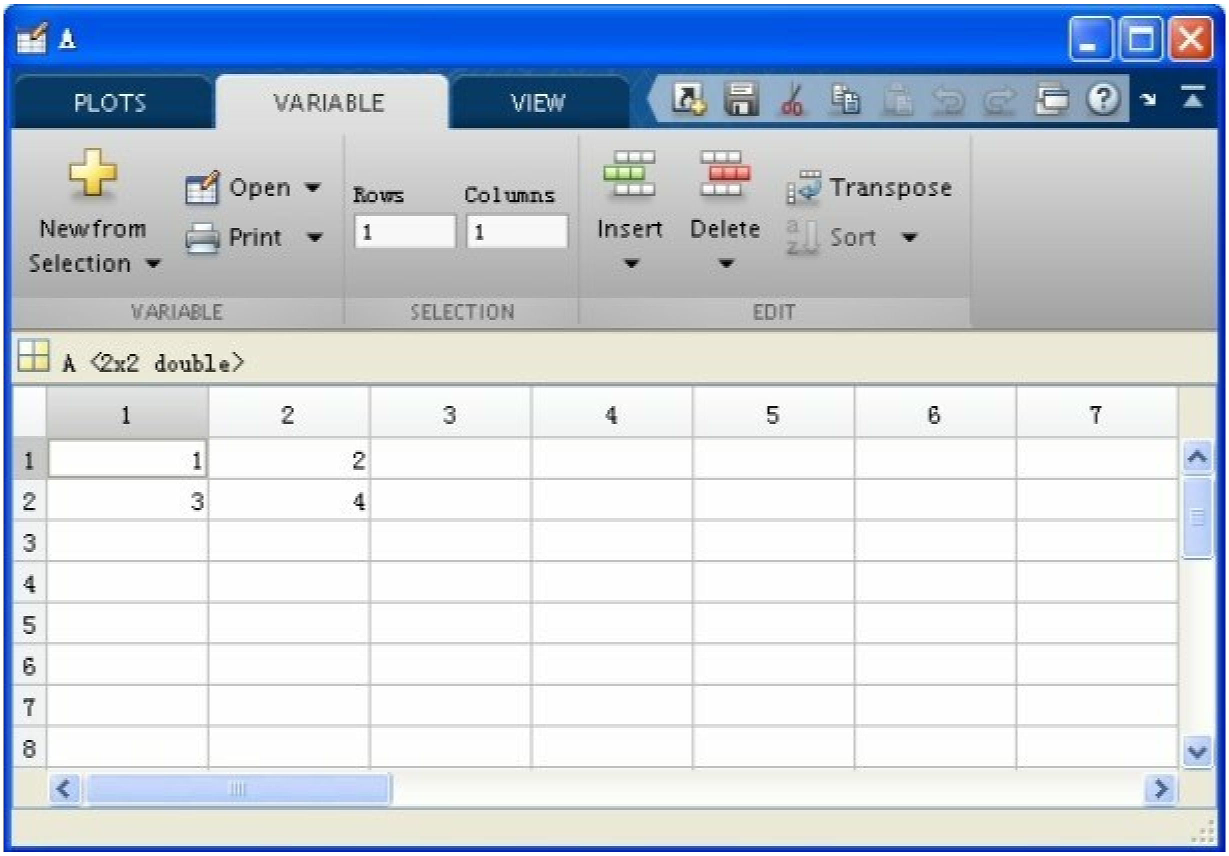


图2-2 变量A的Array Editor窗口

通过鼠标双击Workspace窗口下的2行2列的数组变量A，打开数组A的编辑器界面，如图2-2所示。

在其他位置单击鼠标或者按Enter键，都可以使当前扩展操作即刻生效。如图2-3所示，在第6行第6列的位置单击单元格并输入数值2，数组A被扩展为6行6列的数组，原有元素不变，在第6行第6列的位置赋值为2，其他扩展的位置上元素被默认赋值为0。

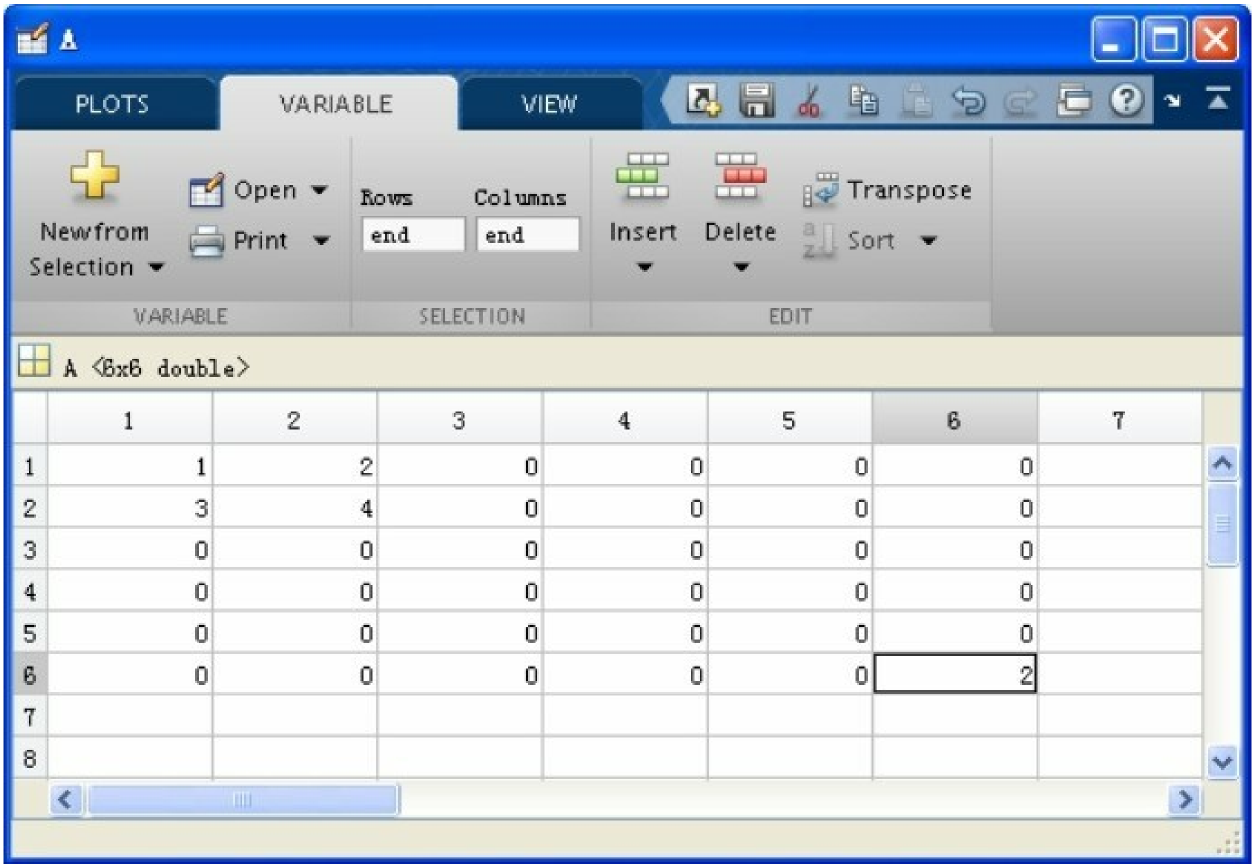


图2-3 Array Editor中扩展后的数组

通过Array Editor也可以裁减数组，这主要是对数组行、列的增加或删除操作，该操作需要通过鼠标右键菜单来实现。在Array Editor 中单击某单元格后，单击鼠标右键，弹出图2-4所示菜单。

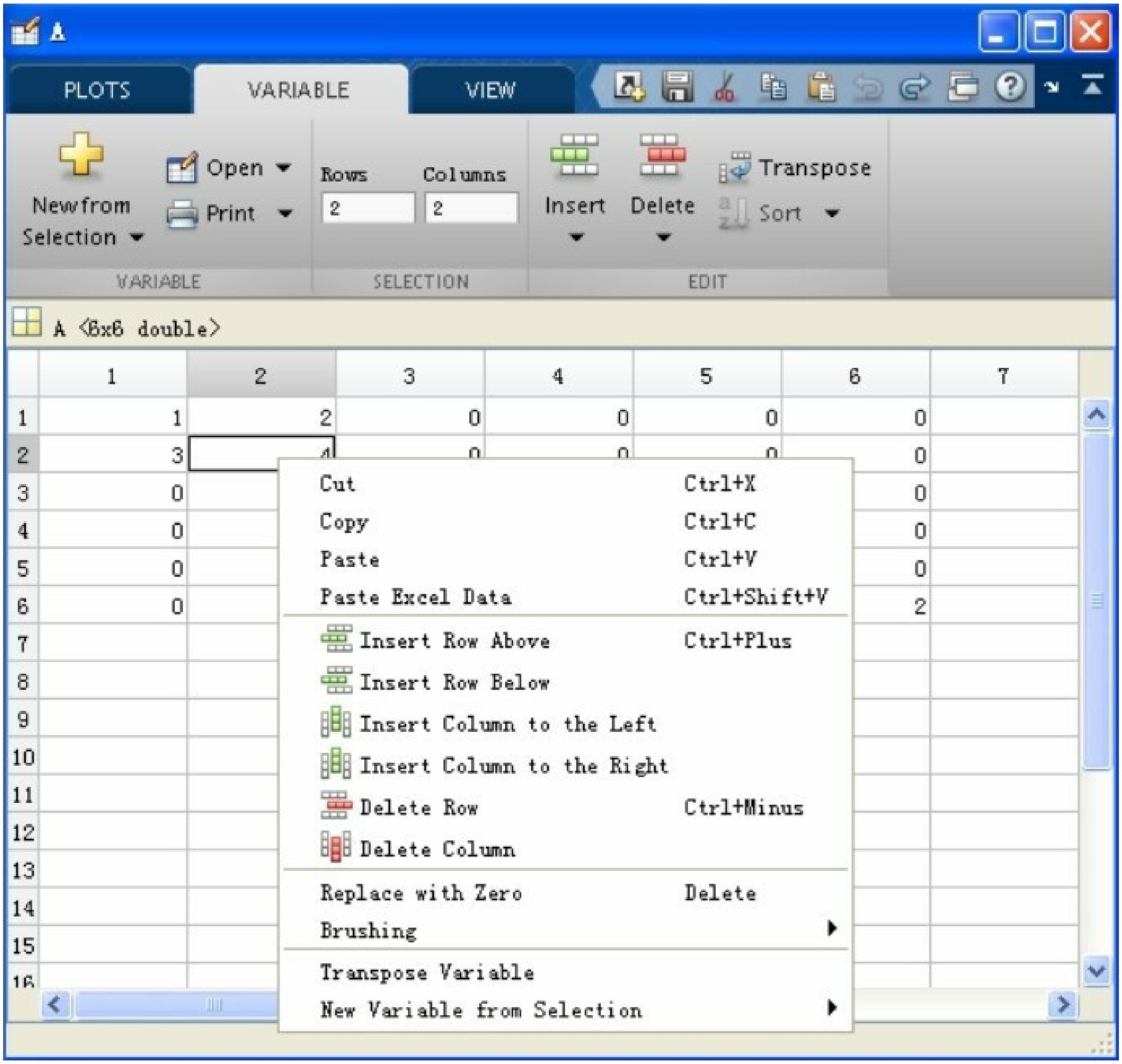


图2-4 Array Editor右键菜单

在图2-4所示的菜单中，选择删除子菜单“Delete…”，就可以指定删除当前数组中选定位置元素所在的整行或者整列。图2-5所示为删除第2行数据之后的数组A。

提示：如果想删除数组中单独一个值，可以选择图2-4中的“Cut”命令。

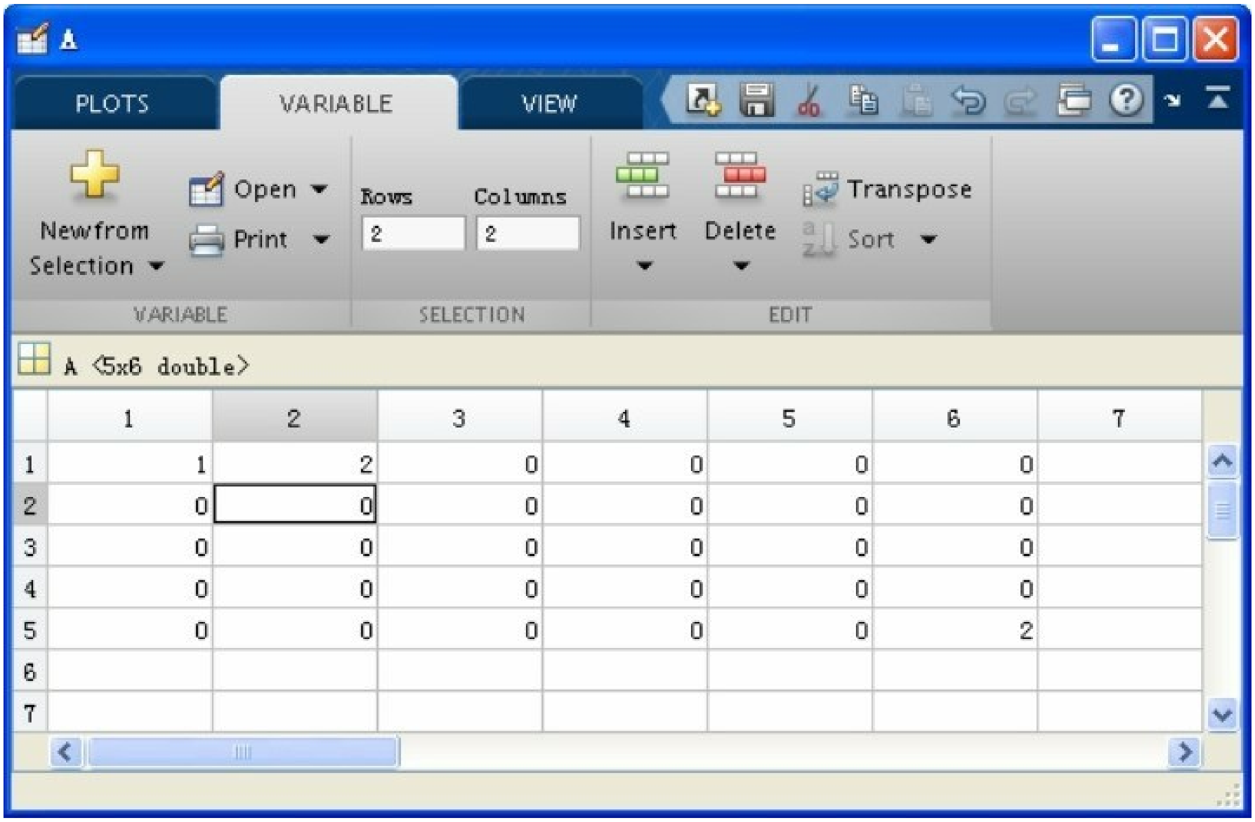


图2-5 删除第2行之后的数组A

图形用户界面的Array Editor使用简单，但如果对数组的扩展或裁剪操作实际比较复杂时，通过Array Editor实现是比较繁琐低效的。

2.数组扩展的cat函数

MATLAB中可以通过cat系列函数将多个小尺寸数组按照指定的连接方式，组合成大尺寸的数组。这些函数包括：cat，horzcat和vertcat。

cat函数可以按照指定的方向将多个数组连接成大尺寸数组。其基本语法格式如下。

C=cat(dim,A1,A2,A3,A4,…)，dim用于指定连接方向，对于两个数组的连接，cat(1,A,B)实际上相当于[A;B]，近似于把两个数组当作两个列元素连接。

horzcat(A1,A2,…)是水平方向连接数组，相当于 cat(A1,A2,…)；vercat(A1,A2,…)是垂直方向连接数组，相当于cat(1,A1,A2,…)。

不管哪个连接函数，都必须保证被操作的数组可以被连接，即在某

个方向上尺寸一致，如horzcat函数要求被连接的所有数组都具有相同的行数，而vercat函数要求被连接的所有数组都具有相同的列数。

【例2-8】通过cat函数扩展数组。

解：在Commond Window窗口输入以下命令。

```
>> A=rand(3,5)
A =
   0.1966    0.4733    0.5853    0.2858    0.3804
   0.2511    0.3517    0.5497    0.7572    0.5678
   0.6160    0.8308    0.9172    0.7537    0.0759
>> B=eye(3)
B =
   1    0    0
   0    1    0
   0    0    1
>> C=magic(5)
C =
   17    24    1    8    15
   23    5    7    14    16
   4    6    13    20    22
   10    12    19    21    3
   11    18    25    2    9
>> cat(1,A,B)            %列数不同，不能垂直连接
Error using cat
CAT arguments dimensions are not consistent.
>> cat(2,A,B)            %行数相同，可以水平连接
ans =
   0.1966    0.4733    0.5853    0.2858    0.3804    1.0000         0         0
```

```
    0.2511    0.3517    0.5497    0.7572    0.5678         0    1.0000         0
    0.6160    0.8308    0.9172    0.7537    0.0759         0         0    1.0000
```

3.块操作函数

MATLAB中还有通过块操作实现数组扩展的函数。

（1）数组块状赋值函数repmat。

repmat(A,m,n)可以将a行b列的元素A当作“单个元素”，扩展出m行n列个由此“单个元素”组成的扩展数组，实际上新产生的数组共有m·a行，n·b 列。

【例2-9】使用块状复制函数repmat。

解：在Commond Window窗口输入以下命令。

```
>> A=eye(2)
A =
    1    0
    0    1
>> repmat(A,2,2)
ans =
    1    0    1    0
    0    1    0    1
    1    0    1    0
    0    1    0    1
```

（2）对角块生成函数blkdiag。

blkdiag（A,B,…）将数组A，B等当作“单个元素”，安排在新数组的主对角位置，其他位置用零数组块进行填充。

【例2-10】使用对角块生成函数blkdiag。

解：在Commond Window窗口输入以下命令。

```
>> A=eye(2)
A =
```

```
    1    0
    0    1
>> B=ones(2,3)
B =
    1    1    1
    1    1    1
>> blkdiag(A,B)
ans =
    1    0    0    0    0
    0    1    0    0    0
    0    0    1    1    1
    0    0    1    1    1
```

（3）块操作函数kron。

kron(X,Y)把数组Y当作一个“元素块”，先复制扩展出size(X)规模的元素块，然后每一个块元素与X的相应位置的元素值相乘。

例如，对 2 行 3 列的数组 X 和任意数组 Y，kron(X,Y)返回的数组相当于[X(1,1)·Y X(1,2)·Y X(1,3)·Y;X(1,3)·Y X(2,2)·Y X(2,3)·Y]。

【例2-11】使用块操作函数kron。

解：在Commond Window窗口输入以下命令。

```
>> A=[0 1;1 2]
B=magic(2)
C=kron(A,B)
A =
    0    1
    1    2
B =
    1    3
```

```
    4    2
C =
    0    0    1    3
    0    0    4    2
    1    3    2    6
    4    2    8    4
```

4.索引扩展

索引扩展是对数组进行扩展中最常用也最易用的方法。前面讲到索引寻址时，其中的数字索引有一定的范围限制，比如m行n列的数组A，要索引寻址访问一个已有元素，通过单下标索引A(a)访问就要求a<=m，b<=n，因为A只有m行n列。

但索引扩展中使用的索引数字，就没有这些限制；相反，必然要用超出上述限制的索引数字，来指定当前数组尺寸外的一个位置，并对其进行赋值，以完成扩展操作。

通过索引扩展，一条语句只能增加一个元素，并同时在未指定的新添位置上默认赋值为 0，因此，要扩展多个元素就需要组合运用多条索引扩展语句，并且经常也要通过索引寻址修改特定位置上被默认赋值为0的元素。

【例2-12】索引扩展。

解：在Commond Window窗口输入以下命令。

```
>> A=eye(3)
A =
    1    0    0
    0    1    0
    0    0    1
>>A(4,6)=25              %索引扩展
A =
```

1 0 0 0 0 0

0 1 0 0 0 0

0 0 1 0 0 0

0 0 0 0 0 25

通过例2-12可见，组合应用索引扩展和索引寻址重新赋值命令，在数组的索引扩展中是经常会遇到的。

5.通过冒号操作符裁剪数组

相对于数组扩展这种放大操作，数组的裁剪就是产生新的子数组的缩小操作，从已知的大数据集中挑出一个子集合，作为新的操作对象，这在各种应用领域都是常见的。

MATLAB中裁剪数组，最常用的就是冒号操作符，实际上，冒号操作符实现裁剪功能时，其意义和冒号用于创建一维数组的意义是一样的，都是实现一个递变效果。

例如，从100行100列的数组A中挑选偶数行偶数列的元素，相对位置不变的组成50行50列的新数组B，只需要通过B=A(2:2:100,2:2:100)就可以实现，实际上这是通过数组数字索引实现了部分数据的访问。

更一般的裁剪语法如下。

B=A([a1,a2,a3,…], [b1,b2,b3,…])

表示提取数组A的a1,a2,a3,…等行，b1,b2,b3,…等列的元素组成子数组B。

此外，冒号还有一个特别的用法。当通过数字索引访问数组元素时，如果某一索引位置上不是用数字表示，而是用冒号代替，则表示这一索引位置可以取所有可以取到的值。例如对5行3列的数组A，A(3,:)表示取A的第三行所有元素（从第1行到第3列），A(:,2)表示取A的第二列的所有元素（从第1行到第5行）。

【例2-13】数组裁剪。

解：在Commond Window窗口输入以下命令。

```
>> A=magic(8)
A =
   64    2    3   61   60    6    7   57
    9   55   54   12   13   51   50   16
   17   47   46   20   21   43   42   24
   40   26   27   37   36   30   31   33
   32   34   35   29   28   38   39   25
   41   23   22   44   45   19   18   48
   49   15   14   52   53   11   10   56
    8   58   59    5    4   62   63    1
>>A(1:3:5,3:7)        %提取数组A的第1、3、5行，3到7列的所有元素
ans =
    3   61   60    6    7
   46   20   21   43   42
   35   29   28   38   39
```

6.数组元素删除

通过部分删除数组元素，也可以实现数组的裁剪，删除数组元素很简单，只需要对该位置元素赋值为空方括号（[]）即可。一般配合冒号，将数组的某些行、列元素删除。但是注意，进行删除时，索引结果必须是完整的行或完整的列，而不能是数组内部的块或单元格。

【例2-14】数组元素删除。

解：在Commond Window窗口输入以下命令。

```
>> A=magic(7)
A =
   30   39   48    1   10   19   28
   38   47    7    9   18   27   29
   46    6    8   17   26   35   37
```

```
    5   14   16   25   34   36   45
   13   15   24   33   42   44    4
   21   23   32   41   43    3   12
   22   31   40   49    2   11   20
>> A(1:3:8,:)=[]
A =
   38   47    7    9   18   27   29
   46    6    8   17   26   35   37
   13   15   24   33   42   44    4
   21   23   32   41   43    3   12
```

通过例2-14可见，数组元素的部分删除是直接在原始数组上进行的操作，在实际应用中，要考虑在数组元素删除前要不要先保存一个原始数组的拷贝，避免不小心造成对原始数据的破坏。另外，单独的一次删除操作只能删除某些行或某些列，因此一般需要通过两条语句才能实现行列两个方向的数组元素删除。

MATLAB中有大量内部函数可以对数组进行改变形状的操作，包括数组转置，数组平移和旋转，以及数组尺寸的重新调整。

2.2.4 数组形状的改变

MATLAB中有大量内部函数可以对数组进行改变形状的操作，包括数组转置，数组平移和旋转，以及数组尺寸的重新调整。

1.数组的转置

MATLAB中进行数组转置最简单的是通过转置操作符（’）。需要注意以下几个方面。

（1）对于有复数元素的数组，转置操作符（’）在变化数组形状的同时，也会将复数元素转化为其共轭复数。

（2）如果要对复数数组进行非共轭转置，可以通过点转置操作符（.’）实现。

（3）共轭和非共轭转置也可以通过 MATLAB 函数完成，transpose实现非共轭转置，功能等同于点转置操作符（.’）：ctranspose实现共轭转置，功能等同于转置操作符（’）。当然，这4种方法对于实数数组转置结果是一样的。

【例2-15】数组转置。

解：在Commond Window窗口输入以下命令。

```
>> A=rand(2,4)
A =
   0.9575    0.1576    0.9572    0.8003
   0.9649    0.9706    0.4854    0.1419
>> A'
ans =
   0.9575    0.9649
   0.1576    0.9706
   0.9572    0.4854
   0.8003    0.1419
>> B=[2-i,3+4i,2,5i;6+i,4-i,2i,7]
B =
   2.0000 -1.0000i    3.0000+ 4.0000i    2.0000              0+ 5.0000i
   6.0000+ 1.0000i    4.0000 -1.0000i    0+ 2.0000i       7.0000
>> B'
ans =
   2.0000+ 1.0000i    6.0000 -1.0000i
   3.0000 -4.0000i    4.0000+ 1.0000i
   2.0000                  0 -2.0000i
```

```
   0 - 5.0000i    7.0000
>> B.'
ans =
   2.0000 -1.0000i    6.0000+ 1.0000i
   3.0000+ 4.0000i    4.0000 -1.0000i
   2.0000                  0+ 2.0000i
   0+ 5.0000i    7.0000
>> transpose(B)
ans =
   2.0000 -1.0000i    6.0000+ 1.0000i
   3.0000+ 4.0000i    4.0000 -1.0000i
   2.0000                  0+ 2.0000i
   0+ 5.0000i    7.0000
```

实际使用中，由于操作符的简便性，经常会使用操作符而不是转置函数来实现转置。但是在复杂的嵌套运算中，转置函数可能是唯一的可用方法。所以，两类转置方式读者都要掌握。

2.数组翻转

MATLAB中数组翻转的函数见表2-7。

表2-7 数组翻转函数

函数及语法	说　　明
fliplr(*A*)	左右翻转数组 *A*
flipud(*A*)	上下翻转数组 *A*
flipdim(*A*,*k*)	按 *k* 指定的方向翻转数组 对于二维数组，*k*=1 相当于 flipud(*A*); *k*=2 相当于 fliplr(*A*)
rot90(*A*,*k*)	把 *A* 逆时针旋转 *k*+90 度，*k* 不指定时默认为 1

【例2-16】数组翻转。

解：在Commond Window窗口输入以下命令。

```
>> A=rand(4,6)
A =
    0.4218    0.6557    0.6787    0.6555    0.2769    0.6948
    0.9157    0.0357    0.7577    0.1712    0.0462    0.3171
    0.7922    0.8491    0.7431    0.7060    0.0971    0.9502
    0.9595    0.9340    0.3922    0.0318    0.8235    0.0344
>> flipud(A)
ans =
    0.9595    0.9340    0.3922    0.0318    0.8235    0.0344
    0.7922    0.8491    0.7431    0.7060    0.0971    0.9502
    0.9157    0.0357    0.7577    0.1712    0.0462    0.3171
    0.4218    0.6557    0.6787    0.6555    0.2769    0.6948
>> fliplr(A)
ans =
    0.6948    0.2769    0.6555    0.6787    0.6557    0.4218
    0.3171    0.0462    0.1712    0.7577    0.0357    0.9157
    0.9502    0.0971    0.7060    0.7431    0.8491    0.7922
    0.0344    0.8235    0.0318    0.3922    0.9340    0.9595
>> flipdim(A,2)
ans =
    0.6948    0.2769    0.6555    0.6787    0.6557    0.4218
    0.3171    0.0462    0.1712    0.7577    0.0357    0.9157
    0.9502    0.0971    0.7060    0.7431    0.8491    0.7922
    0.0344    0.8235    0.0318    0.3922    0.9340    0.9595
>> rot90(A,2)
ans =
    0.0344    0.8235    0.0318    0.3922    0.9340    0.9595
```

```
    0.9502    0.0971    0.7060    0.7431    0.8491    0.7922
    0.3171    0.0462    0.1712    0.7577    0.0357    0.9157
    0.6948    0.2769    0.6555    0.6787    0.6557    0.4218
>> rot90(A)
ans =
    0.6948    0.3171    0.9502    0.0344
    0.2769    0.0462    0.0971    0.8235
    0.6555    0.1712    0.7060    0.0318
    0.6787    0.7577    0.7431    0.3922
    0.6557    0.0357    0.8491    0.9340
    0.4218    0.9157    0.7922    0.9595
```

3.数组尺寸调整

改变数组形状，还有一个常用的函数 reshape，它可以把已知数组改变成指定的行列尺寸。

对于m行n列的数组A，B=reshape(A,a,b)可以将其调整为a行b列的尺寸，并赋值为变量B，这里必须满足m·n=a·b。在尺寸调整前后，两个数组的单下标索引不变，即A(x)必然等于B(x)，只要x是符合取值范围要求的单下标数字。也就是说，按照列优先原则把A和B的元素排列成一列，那结果必然是一样的。

【例2-17】数组尺寸调整。

解：在Commond Window窗口输入以下命令。

```
>> A=rand(3,4)
A =
    0.4387    0.7952    0.4456    0.7547
    0.3816    0.1869    0.6463    0.2760
    0.7655    0.4898    0.7094    0.6797
>> reshape(A,2,6)
```

```
ans =
   0.4387    0.7655    0.1869    0.4456    0.7094    0.2760
   0.3816    0.7952    0.4898    0.6463    0.7547    0.6797
>> reshape(A,2,8)              %a*b不等于m*n时会报错
Error using reshape
To RESHAPE the number of elements must not change.
```

2.2.5 数组运算

本节介绍数组的各种数学运算。

1.数组—数组运算

最基本的就是数组和数组的加（+）、减（−）、乘（*）、乘方（^）等运算。要注意，数组的加、减，要求参与运算的两个数组具有相同的尺寸，而数组的乘法要求第一个数组的列数等于第二个数组的行数。

乘方运算在指数n为自然数时相当于n次自乘，这要求数组具有相同的行数和列数。关于指数为其他情况的乘方，本节不作讨论，读者可以参考有关高等代数书籍。

【例2-18】使用数组—数组运算。

解：在Commond Window窗口输入以下命令。

```
>> A=magic(4)
A =
   16    2    3    13
   5    11    10    8
   9    7    6    12
   4    14    15    1
>> B=eye(4)
```

```
B =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1
>> A+B
ans =
    17    2    3    13
    5    12    10    8
    9    7    7    12
    4    14    15    2
```

数组除法实际上是乘法的逆运算，相当于参与运算的一个数组和另一个数组的逆（或伪逆）数组相乘。MATLAB中数组除法有左除（/）和右除（\）两种。

·A/B 相当于A·inv(B)或A·pinv(B)。

·A\B 相当于inv(A)·B或pinv(A)·B。

其中inv是数组求逆函数，仅适用于行列数相同的方形数组（线性代数中，称为方阵）；pinv 是求数组广义逆的函数。关于逆矩阵和广义逆矩阵的知识，请读者参考有关高等代数书籍。

【例2-19】使用数组除法。

解：在Commond Window窗口输入以下命令。

```
>> A=[3 5 6;2 1 4;2 5 6]
A =
    3    5    6
    2    1    4
    2    5    6
>> B=randn(3)
```

B =

0.5377   0.8622   -0.4336

1.8339   0.3188   0.3426

-2.2588   -1.3077   3.5784

>> A/B

ans =

8.8511   2.1711   2.5413

2.2919   1.9120   1.2125

9.1861   1.6402   2.6328

>> A*inv(B)

ans =

8.8511   2.1711   2.5413

2.2919   1.9120   1.2125

9.1861   1.6402   2.6328

>> pinv(A)*B

ans =

2.7965   2.1699   -4.0120

-0.6323   0.1097   -0.2707

-0.7817   -1.0327   2.1593

2.点运算

前面讲到的数组乘、除、乘方运算，都是专门针对数组定义的运算。在有些情况下，用户可能希望对两个尺寸相同的数组进行元素对元素的乘、除，或者对数组的逐个元素进行乘方，这就可以通过点运算实现。

A.*B，就可以实现两个同样尺寸的数组A和数组B对于元素的乘法，同样地，A./B或A.\B实现元素对元素的除法，A.^n实现对逐个元素的乘方。

【例2-20】使用点运算。

解：在Commond Window窗口输入以下命令。

```
>> A=magic(4)
A =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
>> B=ones(4)+4*eye(4)
B =
    5    1    1    1
    1    5    1    1
    1    1    5    1
    1    1    1    5
>> A.*B
ans =
   80    2    3   13
    5   55   10    8
    9    7   30   12
    4   14   15    5
>>B.*A        %对应的元素的乘法，因此和A.*B结果一样
ans =
   80    2    3   13
    5   55   10    8
    9    7   30   12
    4   14   15    5
>>A.\B        %以A的各个元素为分母，B相对应的各个元素为分
```

子，逐个元素作除法

ans =

0.3125   0.5000   0.3333   0.0769

0.2000   0.4545   0.1000   0.1250

0.1111   0.1429   0.8333   0.0833

0.2500   0.0714   0.0667   5.0000

特别要强调的是，许多MATLAB内置的运算函数，如sqrt，exp，log，sin等，都只能对数组进行逐个元素的相应运算。至于专门的数组的开方、指数等运算，都有专门的数组运算函数。

3.专门针对数组的运算函数

MATLAB中，专门针对数组的运算函数一般末尾都以m结尾（m代表matirx），如sqrtm和 expm 等，这些运算都是特别定义的数组运算，不同于针对单个数值的常规数学运算。这几个函数都要求参与运算的数组是行数和列数相等的方形数组。具体的运算方式请参考高等代数方面的书籍。

【例2-21】使用数组运算函数。

解：在Commond Window窗口输入以下命令。

```
>> A=magic(4)
A =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
>> sqrt(A)
ans =
   4.0000    1.4142    1.7321    3.6056
   2.2361    3.3166    3.1623    2.8284
```

3.0000 2.6458 2.4495 3.4641

2.0000 3.7417 3.8730 1.0000

```
>> sqrtm(A)
ans =
   3.7584 -0.2071i  -0.2271+ 0.4886i  0.3887+ 0.7700i  1.9110 -1.0514i
   0.2745 -0.0130i  2.3243+ 0.0306i  2.0076+ 0.0483i  1.2246 -0.0659i
   1.3918 -0.2331i  1.5060+ 0.5498i  1.4884+ 0.8666i  1.4447 -1.1833i
   0.4063+ 0.4533i  2.2277 -1.0691i  1.9463 - 1.6848i  1.2506+ 2.3006i
>> exp(A)
ans =
   1.0e+06 *
   8.8861 0.0000 0.0000 0.4424
   0.0001 0.0599 0.0220 0.0030
   0.0081 0.0011 0.0004 0.1628
   0.0001 1.2026 3.2690 0.0000
```

2.2.6 数组的查找

MATLAB中数组查找只有一个函数find。它能够查找数组中的非零元素并返回其下标索引。find 配合各种关系运算和逻辑运算，能够实现很多查找功能。find 函数有两种语法形式。

a=find(A)返回数组A中非零元素的单下标索引。

[a,b]=find(A)返回数组A中非零元素的双下标索引方式。

实际应用中，经常通过多重逻辑嵌套产生逻辑数组，判断数组元素是否符合某种比较关系，然后用find函数查找这个逻辑数组中的非零元素，返回符合比较关系的元素的索引，从而实现元素访问。find 用于产生索引数组，过度实现最终的索引访问，因此经常不需要直接指定find函数的返回值。

【例2-22】使用数组查找函数find。

解：在Commond Window窗口输入以下命令。

```
>> A=rand(3,5)
A =
   0.6787    0.3922    0.7060    0.0462    0.6948
   0.7577    0.6555    0.0318    0.0971    0.3171
   0.7431    0.1712    0.2769    0.8235    0.9502
>> A<0.5
ans =
   0    1    0    1    0
   0    0    1    1    1
   0    1    1    0    0
>> A>0.3
ans =
   1    1    1    0    1
   1    1    0    0    1
   1    0    0    1    1
>>   (A>0.3)&(A<0.5)      %逻辑嵌套产生符合多个比较关系的逻辑数组
ans =
   0    1    0    0    0
   0    0    0    0    1
```

0　0　0　0　0

```
>>  find((A>0.3)&(A<0.5))      %逻辑数组中的非零元素，返回符合关系的元素索引
ans =
   4
   14
>>A(find((A>0.3)&(A<0.5)))   %实现元素访问
ans =
   0.3922
   0.3171
```

例2-22分步展示了find最常见用法的具体使用过程。

首先通过rand函数创建了待操作的随机数组A，然后通过比较运算A>0.3和A<0.5返回分别满足某一比较关系的逻辑数组。在这些逻辑数组中，1 代表该位置元素复合比较关系，0则代表不符合比较关系。

然后通过逻辑运算（&）可以产生同时满足两个比较关系的逻辑数组，find 操作这个逻辑数组，返回数组中非零元素的下标索引（本例中返回单下标索引），实际上就是返回原数组中符合两个比较关系的元素的位置索引，然后利用find返回的下标索引就可以寻址访问原来数组中符合比较关系的目标元素。

2.2.7 数组的排序

数组排序也是常用的数组操作，经常用在各种数据分析和处理中，MATLAB中的排序函数是sort。

sort 函数可以对数组按照升序或降序进行排列，并返回排序后的元素在原始数组中的索引位置，sort函数有多种应用语法格式，都有重要的应用，见表2-8。

表2-8 sort函数的各种语法格式

函数语法	说　明
B=sort(A)	对一维或二维数组进行升序排序，并返回排序后的数组 当 A 为二维数组时，则是对数组的每一列进行排序
B=sort(A,dim)	按照设定的方向对数组进行升序排列 dim=1 表示对每一列排序，dim=2 表示对每一行排序

可以看到，sort 都是对单独的一行或一列元素进行排序。即使对于二维数组，也是单独对每一行每一列进行排序，因此返回的索引只是单下标形式，表征排序后的元素在原来行或列中的位置。

【例2-23】数组排序。

解：在Commond Window窗口输入以下命令。

```
>> A=rand(1,8)
A =
   0.0344    0.4387    0.3816    0.7655    0.7952    0.1869    0.4898
0.4456
>> sort(A)          %按照默认的升序方式排列
ans =
   0.0344    0.1869    0.3816    0.4387    0.4456    0.4898    0.7655
0.7952
>> [B,J]=sort(A,'descend')    %降序排列并返回索引
B =
J =
   0.7952    0.7655    0.4898    0.4456    0.4387    0.3816    0.1869
0.0344
   5    4    7    8    2    3    6    1
>>A(J)          %通过索引页可以产生降序排列的数组
ans =
   0.7952    0.7655    0.4898    0.4456    0.4387    0.3816    0.1869
```

0.0344

通过例2-23可见，数组排序函数sort返回的索引，是表示了在排序方向上排序后元素在原数组中的位置。对于一维数组，这就是其单下标索引，但对二维数组，这只是双下标索引中的一个分量，因此不能简单地通过这个返回的索引值寻址产生排序的二维数组。

当然，利用这个索引结果，通过复杂一点的方法也可以得到排序数组，如例2-23中，就可以通过A(J)来产生排序数组，这种索引访问，一般只用在对部分数据的处理上。

2.2.8 高维数组的降维操作

【例2-24】使用squeeze命令来撤销“孤维”，使高维数组进行降维。

解：在Commond Window窗口输入以下命令。

```
>> A=rand(2,3,3)
A(:,:,1) =
   0.1320    0.9561    0.0598
   0.9421    0.5752    0.2348
A(:,:,2) =
   0.3532    0.0154    0.1690
   0.8212    0.0430    0.6491
A(:,:,3) =
   0.7317    0.4509    0.2963
   0.6477    0.5470    0.7447
>>B=cat(4,A(:,:,1),A(:,:,2),A(:,:,3))
B(:,:,1,1) =
   0.1320    0.9561    0.0598
   0.9421    0.5752    0.2348
```

```
B(:,:,1,2) =
    0.3532    0.0154    0.1690
    0.8212    0.0430    0.6491
B(:,:,1,3) =
    0.7317    0.4509    0.2963
    0.6477    0.5470    0.7447
>> C=squeeze(B)
C(:,:,1) =
    0.1320    0.9561    0.0598
    0.9421    0.5752    0.2348
C(:,:,2) =
    0.3532    0.0154    0.1690
    0.8212    0.0430    0.6491
C(:,:,3) =
    0.7317    0.4509    0.2963
    0.6477    0.5470    0.7447
>> size_B=size(B)
size_B =
    2    3    1    3
>> size_C=size(C)
size_C =
    2    3    3
```

2.3 曲线拟合

在科学和工程领域，曲线拟合的主要功能是寻求平滑的曲线来最好地表现带有噪声的测量数据，从这些测量数据中寻求两个函数变量之间的关系或者变化趋势，最后得到曲线拟合的函数表达式y=f(x)。

一般来说，使用多项式进行数据拟合会出现数据振荡，而Spline插值的方法可以得到很好的平滑效果，但是关于该插值方法有太多的参数，不适合曲线拟合的方法。

同时，由于在进行曲线拟合的时候，已经认为所有测量数据中已经包含噪声，因此，最后的拟合曲线并不要求通过每一个已知数据点，衡量拟合数据的标准则是整体数据拟合的误差最小。

一般情况下，MATLAB的曲线拟合方法用的是“最小方差”函数，其中方差的数值是拟合曲线和已知数据之间的垂直距离。

2.3.1 多项式拟合

在MATLAB中，函数polyfit()采用最小二乘法对给定的数据进行多项式拟合，得到该多项式的系数。该函数的调用方式如下。

polyfit(x,y,n)

找到次数为n 的多项式系数，对于数据集合{(xi, yi)}，满足差的平方和最小。

[p,E]=polyfit(x,y,n)

返回同上的多项式P和矩阵E。多项式系数在向量p中，矩阵E用在polyval函数中来计算误差。

【例2-25】某数据的横坐标为x=[0.2 0.3 0.5 0.6 0.8 0.9 1.2 1.3 1.5

1.8]，纵坐标为y=[1 2 3 5 6 7 6 5 4 1]，对该数据进行多项式拟合。

解：代码如下。

```
x=[0.3 0.4 0.7 0.9 1.2 1.9 2.8 3.2 3.7 4.5];
y=[1 2 3 4 5 2 6 9 2 7];
p5=polyfit(x,y,5);          %5阶多项式拟合
y5=polyval(p5,x);
p5=vpa(poly2sym(p5),5)      %显示5阶多项式
p9=polyfit(x,y,9);          %9阶多项式拟合
y9=polyval(p9,x);
figure;                     %画图显示
plot(x,y,'bo');
hold on;
plot(x,y5,'r:');
plot(x,y9,'g--');
legend('原始数据','5阶多项式拟合','9阶多项式拟合');
xlabel('x');
ylabel('y');
```

运行程序后，得到的5阶多项式如下。

```
p5 =
  0.8877*x^5 - 10.3*x^4 + 42.942*x^3 - 77.932*x^2 + 59.833*x - 11.673
```

运行程序后，得到的输出结果如图2-6所示。由图可以看出，使用5次多项式拟合时，得到的结果比较差。

当采用9次多项式拟合时，得到的结果与原始数据符合的比较好。当使用函数polyfit()进行拟合时，多项式的阶次最大不超过length(x)−1。

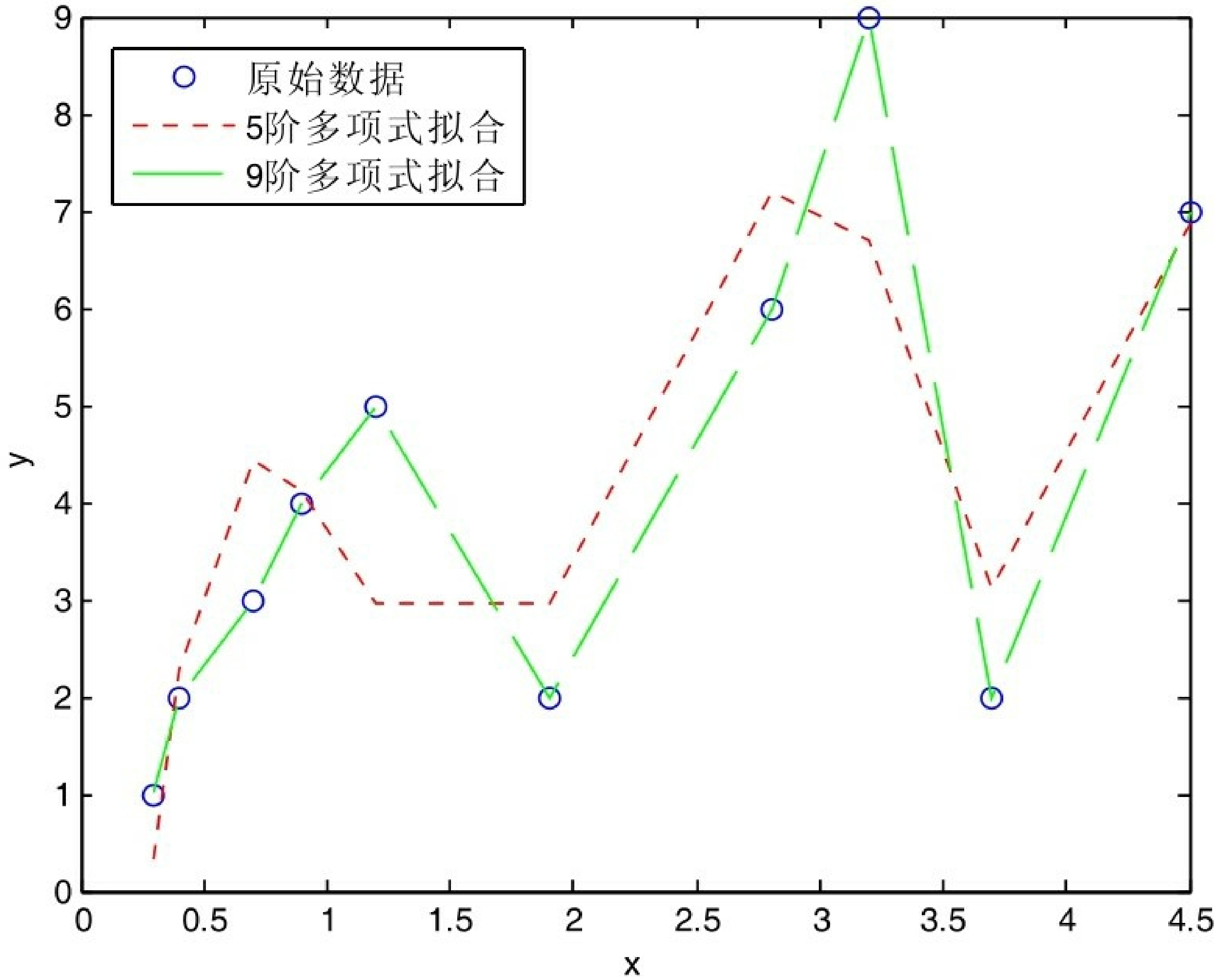


图2-6 多项式曲线拟合

2.3.2 加权最小方差（WLS）拟合原理及实例

所谓加权最小方差，就是根据基础数据本身各自的准确度的不同，在拟合的时候给每个数据以不同的加权数值。这种方法比前面所介绍的单纯最小方差方法要更加符合拟合的初衷。

对应N阶多项式的拟合公式，所需要求解的拟合系数需要求解线性方程组，其中线性方程组的系数矩阵和需要求解的拟合系数矩阵如下。

$$A=\begin{pmatrix} x_1^N & \cdots & x_1 \cdots 1 \\ x_2^N & \cdots & x_2 \cdots 1 \\ \vdots & \ddots & \vdots \\ x_m^N & \cdots & x_m \cdots 1 \end{pmatrix}, \quad \theta=\begin{pmatrix} \theta n \\ \theta n-1 \\ \vdots \\ \theta 1 \end{pmatrix}$$

使用加权最小方差方法求解得到拟合系数

$$\theta_m^n=\begin{pmatrix} \theta^n mn \\ \theta^n mn-1 \\ \vdots \\ \theta^n 1 \end{pmatrix}=\left[A^T MA\right]^{-1} A^T My$$

其对应的加权最小方差为表达式

$$J_m=[A\theta-y]^T W[A\theta-y]$$。

【例2-26】根据WLS数据拟合方法，自行编写使用WLS方法拟合数据的M函数，然后使用WLS方法进行数据拟合。

解：在M文件编辑器中输入下面的程序代码。

```
function [th,err,yi]=polyfits(x,y,N,xi,r)
%x,y为数据点系列，N为多项式拟合的系统，r为加权系数的逆矩阵。
M=length(x);
x=x(:);
y=y(:);
```

```
%判断调用函数的格式
if nargin==4
%当调用函数的格式为（x,y,N,r）
if length(xi)==M
   r=xi;
   xi=x;
%当调用函数的格式为(x,y,N,xi)
else
   r=1;
end
%当调用格式为(x,y,N)
else if nargin==3
   xi=x;
   r=1;
end
%求解系数矩阵
A(:,N+1)=ones(M,1);
for n=N:-1:1
A(:,n)=A(:,n+1).*x;
end
if length(r)==M
   for m=1:M
      A(m,:)=A(m,:)/r(m);
      y(m)=y(m)/r(m);
   end
end
%计算拟合系数
```

```
th=(A\y)';
ye=polyval(th,x);
err=norm(y-ye)/norm(y);
yi=polyval(th,xi);
```

将上面代码保存为“polyfits.m”文件。

使用上面的程序代码，对基础数据进行LS多项式拟合。在MATLAB的命令窗口中输入下面的程序代码。

```
x=[-3:1:3]';
y=[1.1650 0.0751 -0.6965 0.0591 0.6268 0.3516 1.6961]';
[x,i]=sort(x);
y=y(i);
xi=min(x)+[0:100]/100*(max(x)-min(x));
for i=1:4
   N=2*i-1;
   [th,err,yi]=polyfits(x,y,N,xi);
   subplot(2,2,i)
   plot(x,y,'o')
   hold on
   plot(xi,yi,'-')
   grid on
end
```

得到的拟合结果如图2-7所示。

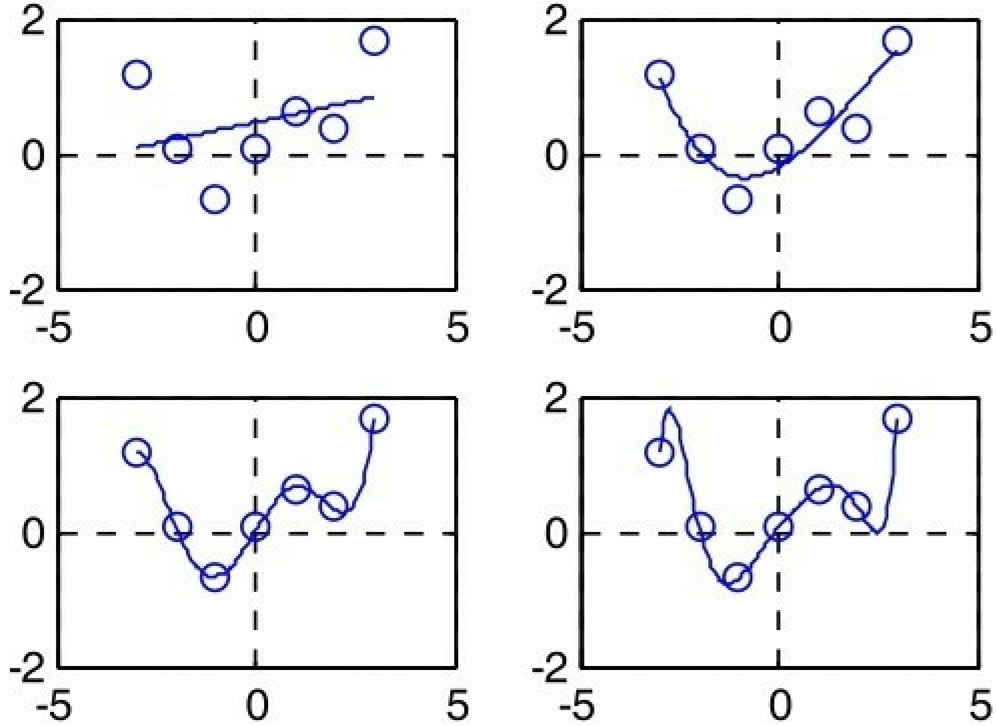


图2-7 使用LS方法求解的拟合结果

从上面的例子可以看出，LS方法其实是WLS方法的一种特例，相当于将每个基础数据的准确度都设为1，但是，自行编写的M文件和默认的命令结果不同，请仔细比较。

2.4 M文件

M 文件有两种形式：脚本文件（Script File）和函数文件（Function File）。脚本文件通常用于执行一系列简单的MATLAB命令，运行时只需输入文件名字，MATLAB就会自动按顺序执行文件中的命令。

函数文件和脚本文件不同，它可以接受参数，也可以返回参数，在一般情况下，用户不能靠单独输入其文件名来运行函数文件，而必须由其他语句来调用，MATLAB的大多数应用程序都以函数文件的形式给出。

2.4.1 M文件概述

MATLAB提供了极其丰富的内部函数，使得用户可以通过命令行调用就可以完成很多工作，但是想要更加高效地利用MATLAB，离不开MATLAB编程。

用户可以通过组织一个MATLAB命令序列完成一个独立的功能，这就是脚本M文件编程；而把M文件抽象封装，形成可以重复利用的功能块，这就是函数M文件编程。因此，MATLAB编程是提高MATLAB应用效率，把MATLAB基本函数扩展为实际的用户应用的必经之道。

M文件是包含MATLAB代码的文件。M文件按其内容和功能可以分为脚本M文件和函数M文件这两大类。

1.脚本M文件

它是许多 MATLAB 代码按顺序组成的命令序列集合，不接受参数的输入和输出，与MATLAB 工作区共享变量空间。脚本 M 文件一般用来实现一个想对独立的功能，比如对某个数据集进行某种分析、绘图，

求解某个已知条件下的微分方程等。用户可以通过在命令窗口直接键入文件名来运行脚本M文件。

通过脚本M文件，用户可以把为实现一个具体功能的一系列MATLAB代码书写在一个M文件中，每次只需要键入文件名即可运行脚本M文件中的所有代码。

2.函数M文件

它也是为了实现一个单独功能的代码块，但它与脚本M文件不同的是需要接受参数输入和输出，函数M文件中的代码一般只处理输入参数传递的数据，并把处理结果作为函数输出参数返回给MATLAB工作区中指定的接受量。

因此，函数M文件具有独立的内部变量空间，在执行函数M文件时，需指定输入参数的实际取值，而且一般要指定接收输出结果的工作区变量。

MATLAB 提供的许多函数就是用函数 M 文件编写的。尤其是各种工具箱中的函数，用户可以打开这些M文件查看。实际上，对应特殊应用领域的用户，如果积累了充足的专业领域应用的函数，就可以组建自己的专业领域工具箱。

通过函数M文件，用户可以把为实现一个抽象功能的MATLAB代码封装成一个函数接口，在以后的应用中重复调用。

MATLAB中的M文件一般包括以下5部分结构。

（1）函数声明行。

这一行只出现在M文件的第一行，通过function关键字表明此文件是一个M文件，并指定函数名、输入和输出参数。

（2）H1行。

这是帮助文字的第一行，给出M文件帮助最关键的信息。当用lookfor查找某个单词相关的函数时，lookfor只在H1行中搜索是否出现指定单词。

（3）帮助文字。

这部分对M文件更加详细的说明，解释M文件实现的功能，M文件中出现的各变量、参数的意义，以及操作版权详细信息等。

（4）M文件正文。

这是M文件实现功能的MATLAB代码部分，通常包括运算、赋值等指令。

（5）注释部分。

这部分出现的位置比较灵活，主要是用来注释M文件正文的具体运行过程，方便阅读和修改，经常穿插在M文件正文中间。

2.4.2 局部变量与全局变量

无论在脚本文件还是在函数文件中，都会定义一些变量。函数文件所定义的变量是局部变量，这些变量独立于其他函数的局部变量和工作空间的变量，即只能在该函数的工作空间引用，而不能在其他函数工作空间和命令工作空间引用。但是如果某些变量被定义成全局变量，就可以在整个 MATLAB 工作空间进行存取和修改，以实现共享。因此，定义全局变量是函数间传递信息的一种手段。

用命令global定义全局变量，其格式如下。

global A B C

将A、B、C这3个变量定义为全局变量。

在M文件中定义全局变量时，如果在当前工作空间已经存在相同的变量，系统将会给出警告，说明由于将该变量定义为全局变量，可能会使变量的值发生改变。为避免发生这种情况，应该在使用变量前先将其定义为全局变量。

提示：在 MATLAB 中对变量名是区分大小写的，因此为了在程序中分清楚而不至于误声明，习惯上可以将全局变量定义为大写字母。

【例2-27】全局变量的使用。

解：在Commond Window窗口输入以下命令。

```
function y=myt(x)
global a;
a=a+9;
y=cos(x);
```

然后在命令窗口声明全局变量赋值调用。

```
>> global a
>> a=2
a =
   2
>> myt(pi)
ans =
   -1
>> cos(pi)
ans =
   -1
>> a
a =
   11
```

通过例2-27可见，用global将a声明为全局变量后，函数内部对a的修改也会直接作用到MATLAB工作区中，函数调用一次后，a的值从2变为11。

2.4.3 M文件的编辑与运行

MATLAB　语言是一种高效的编程语言，可以用普通的文本编辑器

把一系列 MATLAB语句写在一起构成MATLAB程序，然后存储在一个文件里，文件的扩展名为.m，因此称为M文件。这些文件都是由纯ASCII码字符构成的，在运行M文件时只需在MATLAB命令窗口下输入该文件名即可。

在MATLAB的编辑器中建立与编辑M文件的一般步骤如下。

1.新建文件

（1）最简单的方法是单击MATLAB主界面工具栏上的

图标。

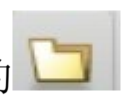
（2）在Commond Window窗口输入edit 语句建立新文件，或输入edit filename 语句，打开名为filename的M文件，在弹出文件不存在的提示框中，单击“Yes”按钮，则建立名为filename 新的M 文件。

（3）选择MATLAB主界面的Home→New子菜单，在其下拉菜单中选择“Script”项。

（4）如果已经打开了文件编辑器后需要再建立新文件，可以用编辑器的菜单或工具栏上相应的图标进行操作。

提示：还可以使用“Ctrl+N”的快捷方式打开一个新的M文件。

2.打开文件

（1）单击MATLAB主界面工具栏上的图标，弹出Open窗口，选择已有的M文件，单击“打开”按钮，如图2-8所示。

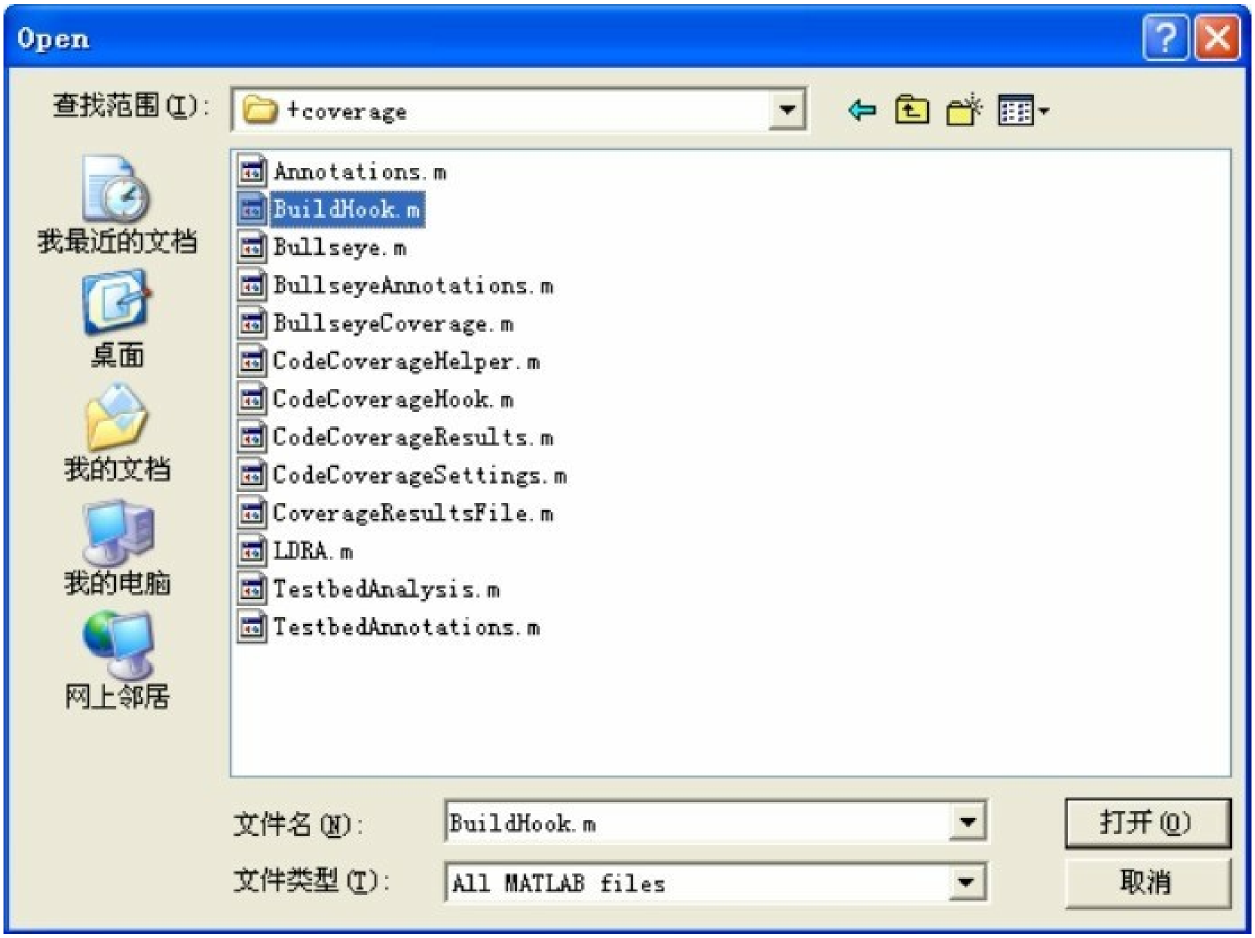


图2-8 Open窗口

（2）输入edit filename 语句，打开名为filename 的M 文件。

3.编辑文件

虽然M文件是普通的文本文件，在任何的文本编辑器中都可以编辑，但MATLAB系统提供了一个更方便的内部编辑器（Editor），如图2-9所示。

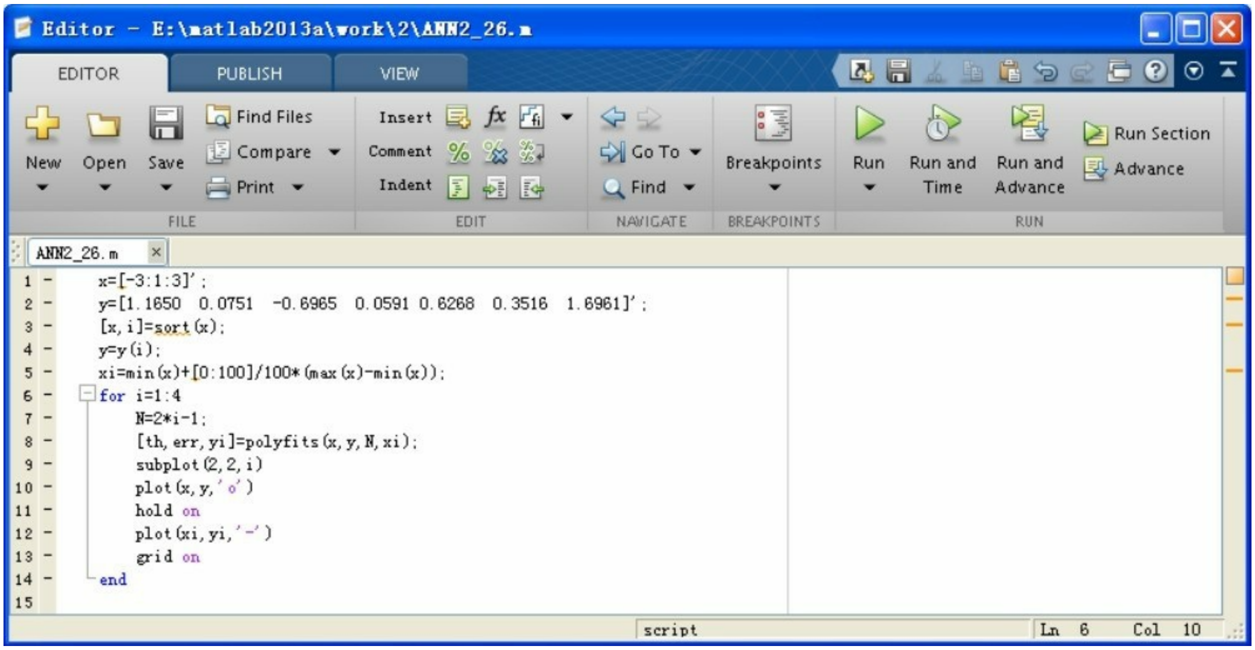


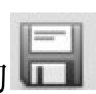
图2-9 Editor窗口

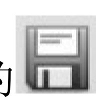
对于新建的M文件，可以在MATLAB的Editor窗口编写新的文件；对于打开的已有M文件，其内容显示在Editor窗口，用户可以对其进行修改。

在编辑的过程中可以使用类似于其他文本编辑器（如Word）的“复制”（Copy）菜单、工具栏的编辑图标和快捷键等，值得注意的是，除了注释内容外，所有MATLAB的语句都要使用西文字符。

4.保存文件

M文件在运行之前必须先保存。其方法如下。

·单击编辑器工具栏上的图标：对于新建的M文件，则弹出“保存文件”（Save file as）对话框，选择存放的路径、文件名和文件保存类型（不选择时为M 文件），单击“保存”按钮，即可完成保存；对于打开的已有M文件，则直接完成保存。

·使用“Ctrl+S”的快捷方式。等同于单击编辑器工具栏上的图标。

·选择工具栏Save 下拉菜单中的“Save As…”选项，可以在弹出“保存文件”（Select file for Save as）对话框中，重新选择存放的目录、文件名进行保存。

5.运行文件

脚本文件可直接运行，而函数文件还必须输入函数参数。

（1）在命令窗口输入要运行的文件名即可开始运行，需要注意的是，在运行前，一定要先保存文件，否则运行的是保存前的程序。

（2）如果在编辑器中完成编辑后需要直接运行，可以直接用鼠标单击图标。

（3）按F5键则保存程序并直接运行；如果是新建M文件，则弹出“保存文件”对话框，用户保存文件后直接运行。

2.4.4 脚本文件

脚本文件是M文件中最简单的一种，不需要输入顿号输出参数，用命令语句可以控制MATLAB命令工作空间的所有数据。

在运行过程中，产生的所有变量均是命令工作空间变量，这些变量一旦生成，就一直保存在内存空间中，除非用户执行clear命令将它们清除。运行一个脚本文件等价于从命令窗口中顺序运行文件里的语句。由于脚本文件只是一串命令的集合，因此只需像在命令窗口中输入语句那样，依次将语句编辑在脚本文件中即可。

【例2-28】编程计算向量元素的平均值。

解：在MATLAB编辑窗口输入以下命令。

```
a=input('输入变量：a=');
[b,c]=size(a);
if～((b==1)|(c==1))|(((b==1)&(c==1)))   %判断输入是否为向量
   error('必须输入向量')
```

```
end
average=sum(a)/length(a)          %计算向量 a所有元素的平均值
```

将其保存为pingjun.m，运行之，如果输入行向量[1 2 3]，则运行结果如下。

```
>> pingjun
输入变量：a=[1 2 3]。
average =
    2
```

如果输入的不是向量，如[1 2; 3 4]，则运行结果如下。

```
>> pingjun
输入变量：a=[1 2; 3 4]，则运行结果如下。
Error using pingjun (line 4)
```

这是因为必须输入向量。

2.4.5 函数文件

如果M文件的第一个可执行语句以function开始，该文件就是函数文件，每一个函数文件都定义一个函数。事实上，MATLAB 提供的函数命令大部分都是由函数文件定义的，这足以说明函数文件的重要。

从使用的角度看，函数是一个“黑箱”，把一些数据送进去，经加工处理，把结果送出来。从形式上看，函数文件区别于脚本文件之处在于脚本文件的变量为命令工作空间变量，在文件执行完成后保留在命令工作空间中；而函数文件内定义的变量为局部变量，只在函数文件内部起作用，当函数文件执行完后，这些内部变量将被清除。

【例2-29】编写函数 average( )用于计算向量元素的平均值。

解：在MATLAB编辑窗口输入以下命令。

```
function y=average(x)
```

```
[a,b]=size(x);              %判断输入量的大小
if～((a==1)|(b==1))| ((a==1)& (b==1))    %判断输入是否为向量
    error('必须输入向量。')
end
y=sum(x)/length(x);           %计算向量x所有元素的平均值
```

将文件存盘，默认状态下函数名为 average_2.m(文件名与函数名相同)，函数 average接受一个输入参数并返回一个输出参数，该函数的用法与其他 MATLAB 函数一样；在MATLAB 命令窗口中运行以下语句，便可求得1～9 的平均值。

```
>> x=1:9
x =
    1    2    3    4    5    6    7    8    9
>> average(x)
ans =
    5
```

通常函数文件由以下几个基本部分组成。

（1）函数定义行。函数定义行由关键字 function 引导，指明这是一个函数文件，并定义函数名、输入参数和输出参数，函数定义行必须为文件的第一个可执行语句，函数名与文件名相同，可以是MATLAB中任何合法的字符。

函数文件可以带有多个输入和输出参数，如下。

```
function [x,y,z]=sphere(theta,phi,rho)
```

也可以没有输出参数，如下。

```
function printresults(x)
```

（2）H1行。H1行就是帮助文本的第一行，是函数定义行下的第一个注释行，是供lookfor查询时使用的。一般来说为了充分利用MATLAB的搜索功能，在编制M文件时，应在H1行中尽可能多地包含该函数的特

征信息。由于在搜索路径上包含 average 的函数很多，因此用lookfor average 语句可能会查询到多个有关的命令。如下。

>> lookfor average_2

average_2.m:% 函数average_2(x)用以计算向量元素的平均值。

（3）帮助文本。在函数定义行后面，连续的注释行不仅可以起到解释与提示作用，更重要的是为用户自己的函数文件建立在线查询信息，以供help命令在线查询时使用。如下。

>> help average_2

函数 average_2(x)用以计算向量元素的平均值。

输入参数x为输入向量，输出参数y为计算的平均值。非向量输入将导致错误。

（4）函数体。函数体包含了全部的用于完成计算及给输出参数赋值等工作的语句，这些语句可以是调用函数、流程控制、交互式输入/输出、计算、赋值、注释和空行。

（5）注释。以%起始到行尾结束的部分为注释部分，MATLAB的注释可以放置在程序的任何位置，可以单独占一行，也可以在一个语句之后，如

%非向量输入将导致错误

[m,n]=size(x);%判断输入量的大小

2.4.6 函数调用

调用函数文件的一般格式如下。

[输出参数表]=函数名（输入参数表）

调用函数时应注意以下几项。

（1）当调用一个函数时，输入和输出参数的顺序应与函数定义时一致，其数目可以按少于函数文件中所规定的输入和输出参数调用函

数，但不能使用多于函数文件所规定的输入和输出参数数目。如果输入和输出参数数目多于或少于函数文件所允许的数目，则调用时自动返回错误信息。举例如下。

```
>> [x,y]=sin(pi)
??? Error using ==> sin
Too many output arguments.
```

又如：

```
>> y=linspace(2)
??? Input argument "n" is undefined.
Error in ==> linspace at 21
```

（2）在编写函数文件调用时常通过nargin、nargout　函数来设置默认输入参数，并决定用户所希望的输出参数。函数nargin　可以检测函数被调用时用户指定的输入参数个数；函数nargout　可以检测函数被调用时用户指定的输出参数个数。

在函数文件中通过nargin、nargout　函数，可以适应函数被调用时，用户输入和输出参数数目少于函数文件中 function 语句所规定数目的情况，以决定采用何种默认输入参数和用户所希望的输出参数。举例如下。

```
function y = linspace(d1, d2, n)
%LINSPACE Linearly spaced vector.
% LINSPACE(X1, X2) generates a row vector of 100 linearly
% equally spaced points between X1 and X2.
%
% LINSPACE(X1, X2, N) generates N points between X1 and X2.
% For N < 2, LINSPACE returns X2.
%
% Class support for inputs X1,X2:
```

```
% float: double, single
%
% See also LOGSPACE, :.
% Copyright 1984-2004 The MathWorks, Inc.
%$Revision: 5.12.4.1 $$Date: 2004/07/05 17:01:20 $
if nargin == 2
    n = 100;
end
n = double(n);
y = [d1+(0:n-2)*(d2-d1)/(floor(n)-1) d2];
```

如果用户只指定2 个输入参数调用linspace，例如linspace(0,10)，linspace 在0～10 之间等间隔产生 100 个数据点；相反，如果输入参数的个数是 3，例如，linspace(0,10,50)，第3 个参数决定数据点的个数，linspace 在0～10 等间隔产生50 个数据点。同样，函数也可按少于函数文件中所规定的输出参数进行调用。例如对函数size( )的调用。

```
>> x=[1 2 3 ; 4 5 6];
>> m=size(x)
m =
    2 3
>> [m,n]=size(x)
m =
    2
n =
    3
```

（3）当函数有一个以上输出参数时，输出参数包含在方括号内。例如，[m,n]=size(x)。注意：[m,n]在左边表示函数的两个输出参数m和n；不要把它和[m,n]在等号右边的情况混淆，如y=[m,n]表示数组y由变

量m和n所组成。

（4）当函数有一个或多个输出参数，但调用时未指定输出参数，则不给输出变量赋任何值。举例如下。

```
function t=toc
% TOC Read the stopwatch timer.
% TOC, by itself, prints the elapsed time (in seconds) since TIC was used.
% t = TOC; saves the elapsed time in t, instead of printing it out.
% See also TIC, ETIME, CLOCK, CPUTIME.
% Copyright(c)1984-94byTheMathWorks,Inc.
% TOC uses ETIME and the value of CLOCK saved by TIC.
Global TICTOC
If nargout<1
elapsed_time=etime(clock,TICTOC)
else
t=etime(clock,TICTOC);
end
```

如果用户调用toc 时不指定输出参数t，举例如下。

```
>> tic
>> toc
elapsed_time =
   4.0160
```

函数在命令窗口显示函数工作空间变量 elapsed_time 的值，但在MATLAB 命令工作空间里不给输出参数t赋任何值，也不创建变量t。

如果用户调用toc时指定输出参数t，举例如下。

```
>> tic
>> out=toc
```

```
out =
   2.8140
```

则以变量out 的形式返回到命令窗口，并在MATLAB 命令工作空间里创建变量out。

（5）函数有自己的独立工作空间，它与MATLAB的工作空间分开。除非使用全局变量，函数内变量与MATLAB其他工作空间之间唯一的联系是函数的输入和输出参数。如果函数任一输入参数值发生变化，其变化仅在函数内出现，不影响 MATLAB 其他工作空间的变量。函数内所创建的变量只驻留在该函数工作空间，而且只在函数执行期间临时存在，以后就消失。因此，从一个调用到另一个调用，在函数工作空间以变量存储信息是不可能的。

（6）在MATLAB其他工作空间重新定义预定义的变量（例如pi），它不会延伸到函数的工作空间；反之亦然，即在函数内重新定义预定义的变量不会延伸到MATLAB的其他工作空间中。

（7）如果变量说明是全局的，函数可以与其他函数、MATLAB的Workspace和递归调用本身共享变量。为了在函数内或MATLAB命令工作空间中访问全局变量，全局变量在每一个所希望的工作空间都必须说明。

（8）全局变量可以为编程带来某些方便，但却破坏了函数对变量的封装，所以在实际编程中，无论什么时候都应尽量避免使用全局变量。如果一定要用全局变量，建议全局变量名要长、采用大写字母，并有选择地以首次出现的M文件的名字开头，使全局变量之间不必要的互作用减至最小。

（9）MATLAB 以搜寻脚本文件的同样方式搜寻函数文件。例如，输入 cow 语句， MATLAB首先认为cow是一个变量；如果它不是，那么MATLAB认为它是一个内置函数；如果还不是，MATLAB 检查当前cow.m 的目录或文件夹；如果仍然不是，MATLAB 就检查cow.m在

MATLAB搜寻路径上的所有目录或文件夹。

（10）从函数文件内可以调用脚本文件。在这种情况下，脚本文件查看函数工作空间，不查看MATLAB的Workspace。从函数文件内调用的脚本文件不必调到内存进行编译，函数每调用一次，它们就被打开和解释。因此，从函数文件内调用脚本文件减慢了函数的执行。

（11）当函数文件到达文件终点，或者碰到返回命令return，就结束执行和返回。返回命令return提供了一种结束函数的简单方法，而不必到达文件的终点。

2.4.7 M文件调试工具

当完成MATLAB代码编写后，用户就可以在命令窗口运行代码（脚本或函数文件）。对于比较简单的代码，一般只要编程习惯较好，都可以一次通过，但对于很多比较复杂的情况，或者用户初学MATLAB编程，一些常见的错误还不能避免，容易在运行时出现错误，这时候，就需要利用MATLAB的调试工具对出现错误的代码进行调试纠错。

MATLAB 的Editor 窗口是一个综合了代码编写、调试的集成开发环境。MATLAB代码调试过程，主要是通过Editor窗口上的工具栏完成，如图2-10所示。



图2-10 Editor窗口上的工具栏

下面介绍Editor窗口上几种经常用到的工具。

（1）Compare：比较两个M文件的差异，并出具报告，报告格式如图2-11所示。

（2）Insert：在M文件中插入注释、函数或者数据等。

（3）Comment：屏蔽或者取消屏蔽M文件中的内容。

（4）Indent：调整M文件内容的位置。

（5）Run：运行当前M文件，快捷键是F5；当前M文件设置了断点时，运行到断点处暂停。

（6）Run Section：运行当前M 文件中用户选中的某一部分程序。

（7）Breakpoints：设置或清除断点。其下拉菜单中的命令如图2-12所示。

当选择该选项下拉菜单中的Set Condition 时，会打开条件断点设置对话框，如图2-13所示，该命令用于设置满足断点有效的条件。

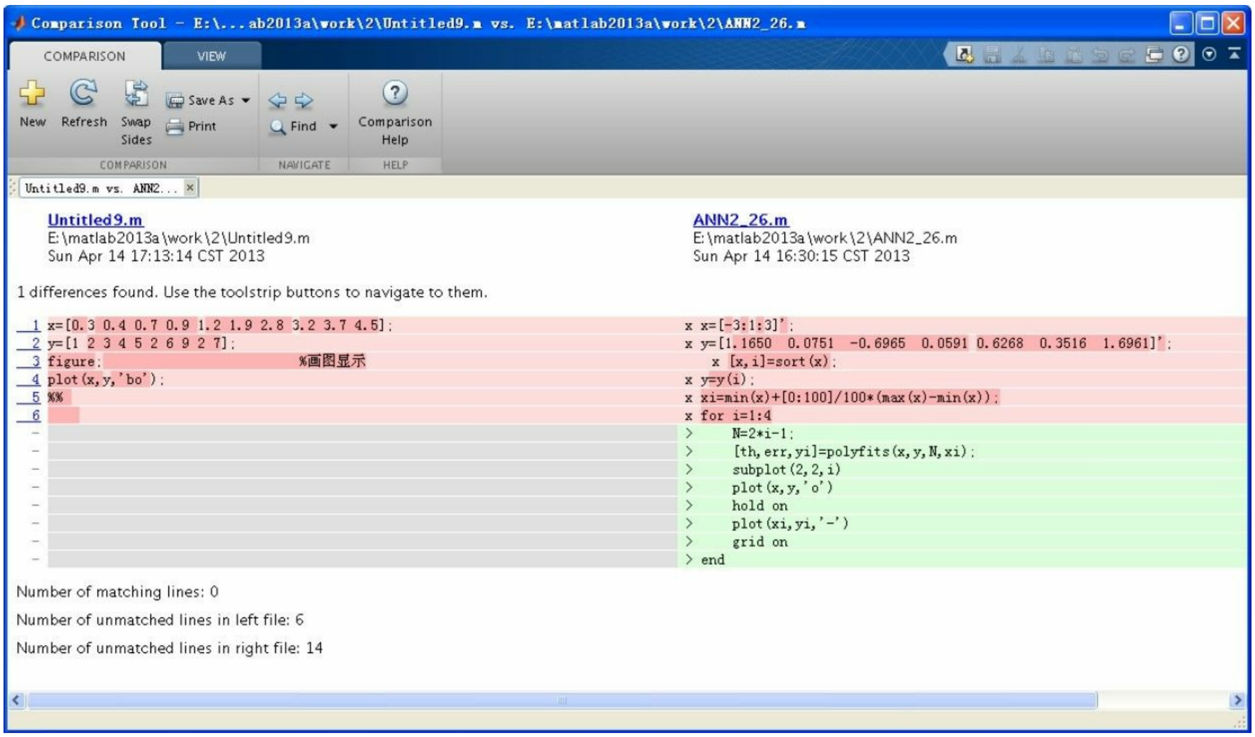


图2-11 两个M文件的比较分析报告

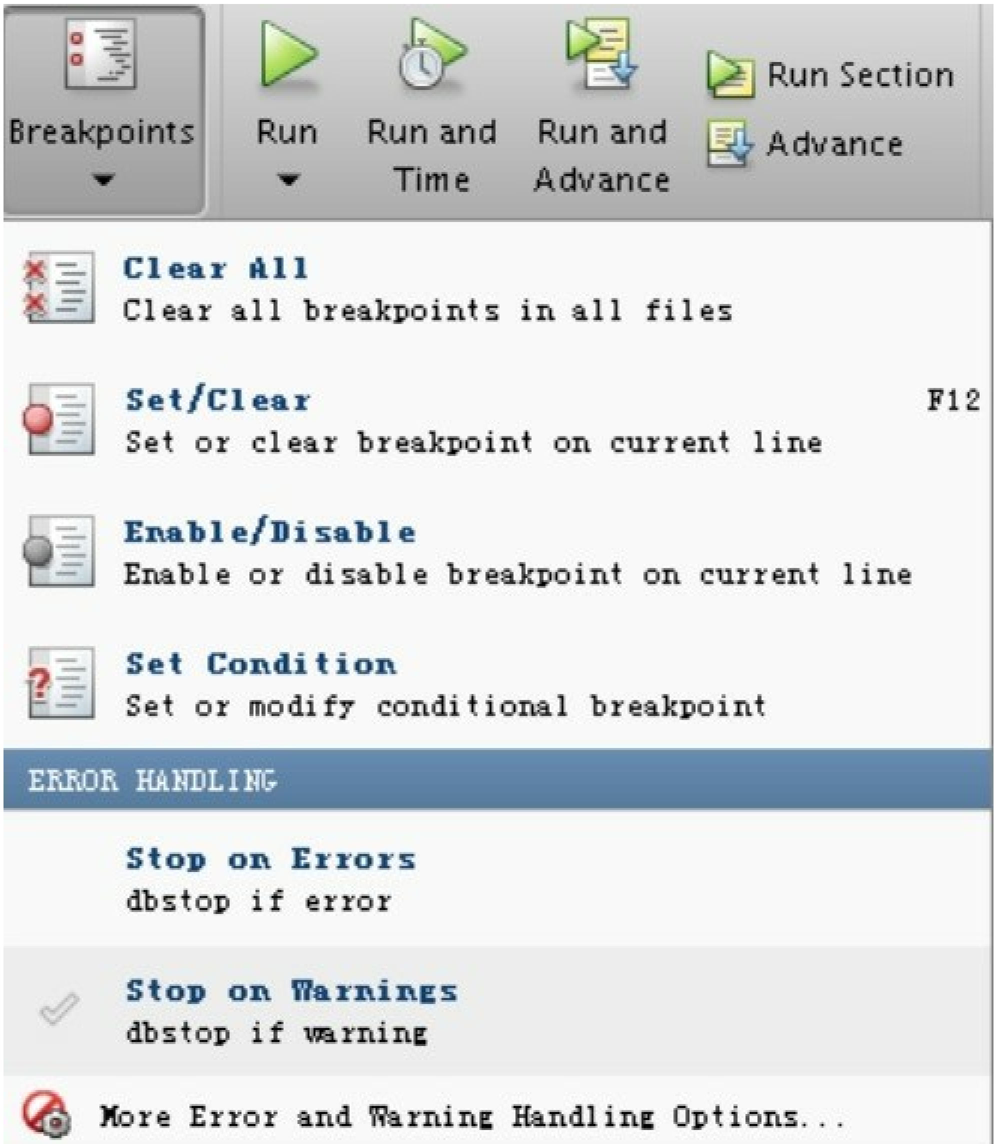


图2-12 Breakpoints下拉菜单选项

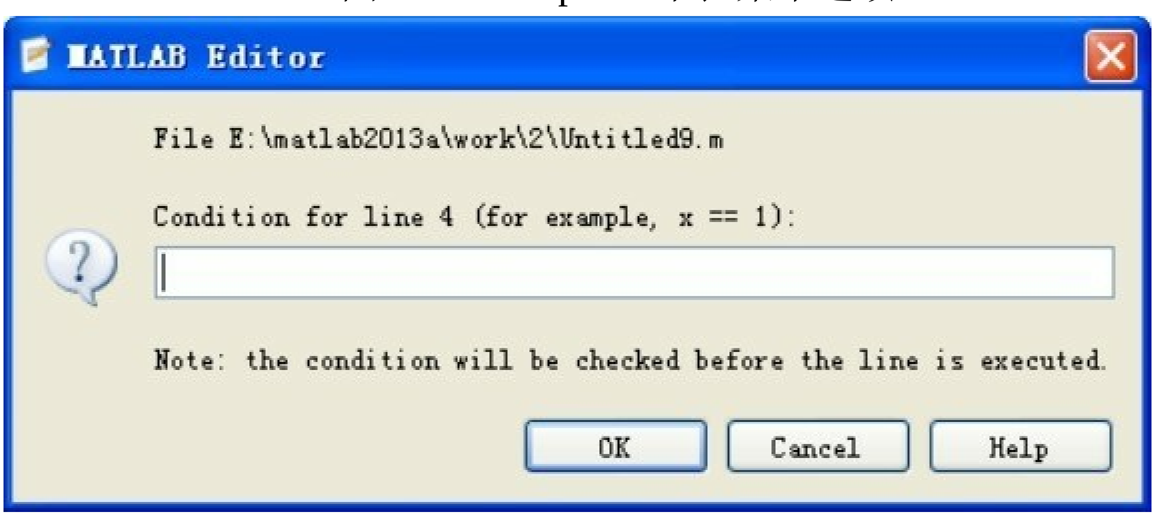


图2-13 条件断点设置对话框

当选择Stop on Errors 或Stop on Warnings 命令时，M 文件会在出现某种运行错误或警告时，停止程序运行。

通常的调试过程是先单击Run按钮，运行一遍M文件，针对系统给出的具体的出错信息，在适当的地方设置断点或条件断点，再次运行到断点位置（如图 2-14 所示），此时MATLAB 把运行控制权交给键盘，命令窗口出现“K>>”提示符（如图 2-15 所示），此时可以在命令窗口查询M文件运行过程中的所有变量，包括函数运行时的中间变量，运行到断点位置后，用户可以逐行运行并实时查询变量取值，从而逐渐找到错误所在并将其排除。

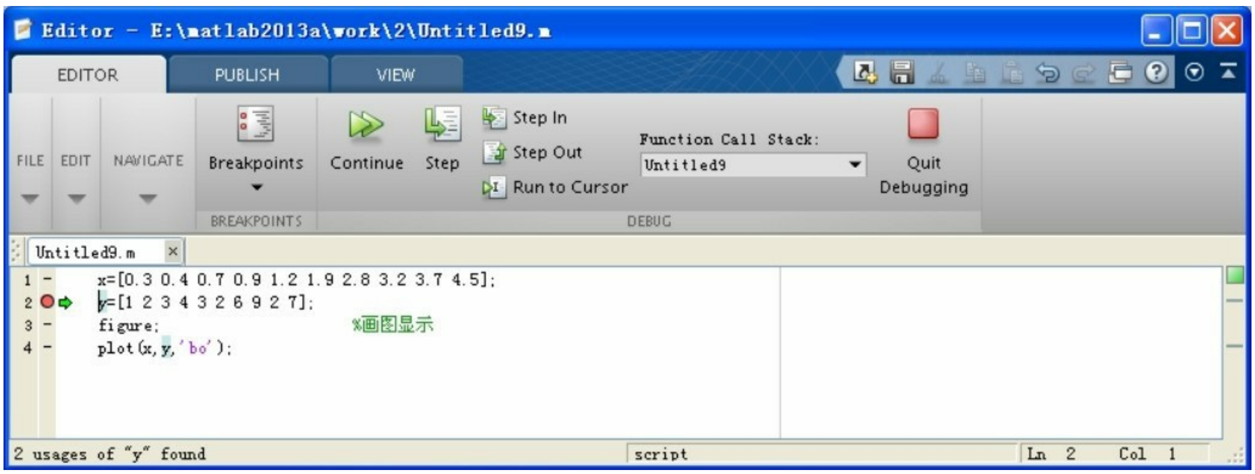


图2-14 设置断点后Run运行到断点所在位置

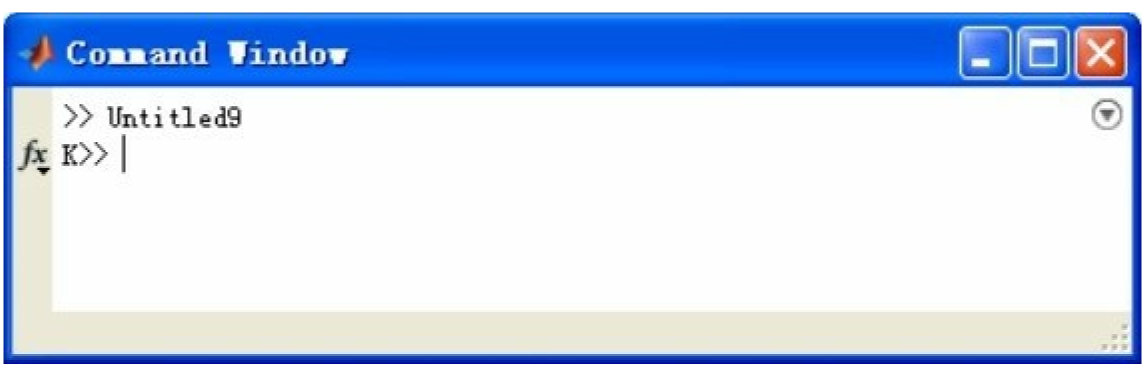


图2-15 调试模式时MATLAB命令窗口把控制权交给键盘

2.4.8 M文件分析工具

通过对M文件进行调试，可以对文件中的编写错误和运行错误进行纠正。完成了调试后，用户编写的M文件就可以正确地运行了，但可能运行效率还不是最优，这就需要通过MATLAB提供的分析工具对代码进行分析，然后有针对性地进行优化。

MATLAB 提供的M 文件分析工具包括Code Analyzer Report工具和Profiler工具，它们都有图形操作界面，使用简单方便，是MATLAB程序分析优化的必用工具。

1.Code Analyzer Report工具

Code Analyzer Report 工具可以分析用户M 文件中的错误或性能问题，用户可以先在代码编辑调试器中打开待分析的M文件，然后选择MATLAB工具菜单下的

，即可得到图2-16所示的代码分析报告。

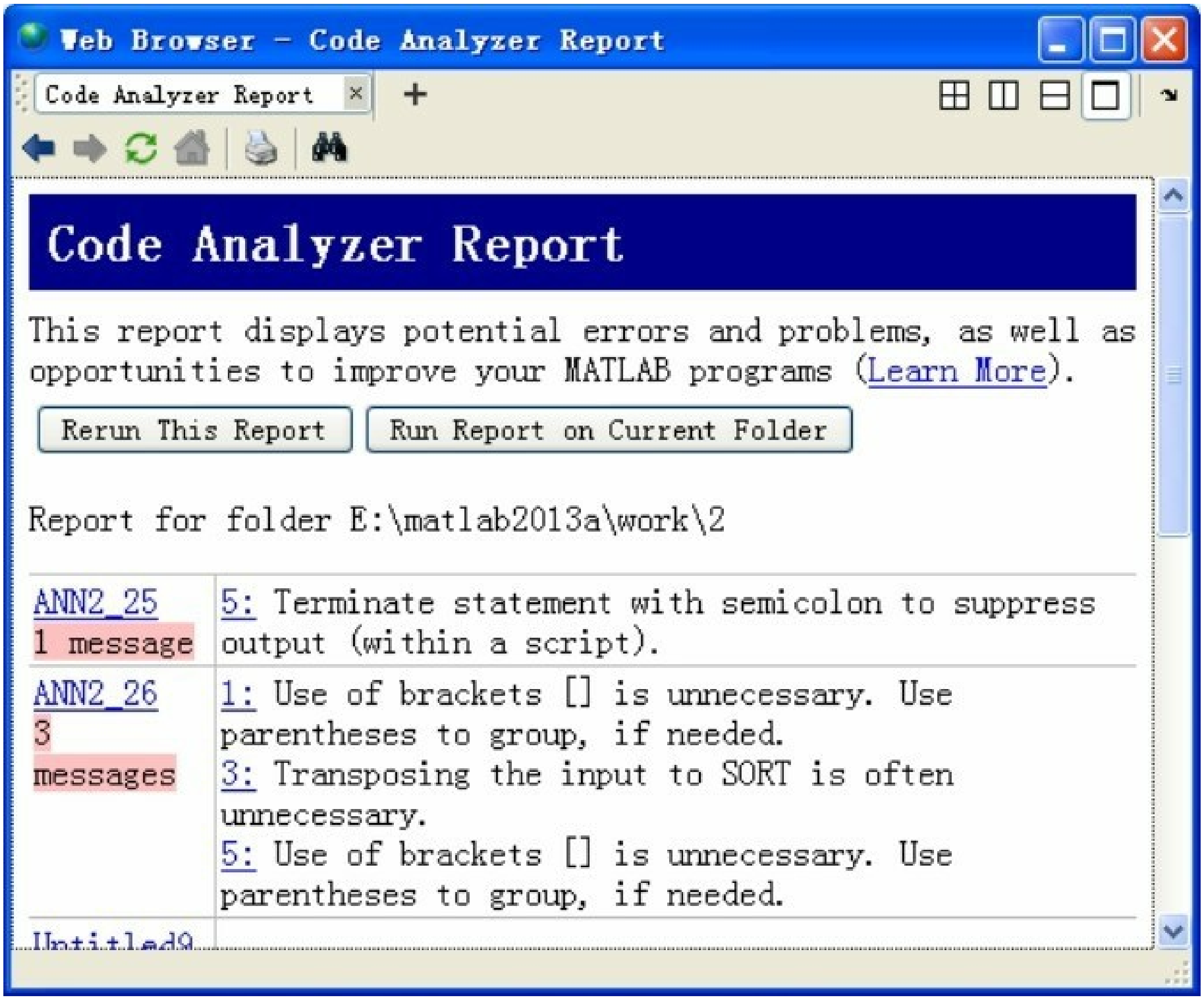


图2-16 分析结果

从图2-16 中可以看出，Code Analyzer Report 分析完成后，会返回一个浏览器界面下的分析报告，报告中包括被分析的M文件的路径，以及若干个分析结果。分析结果的格式是“行号：错误或问题报告”。

实际上，Code Analyzer Report 分析得到的报告并不一定必须要消除，要具体问题具体分析。当用户认可某一条分析结果时，可以单击分析结果中的行号。就可以快捷打开相应的M文件并定位到该行，用户就可以方便地修改代码了。

Code Analyzer Report 不仅可以分析单个M 文件，还可以分析一个文件夹下的所有M文件。在分析结果的主界面下，选择Rerun This

Report 则分析单个M 文件，选择Run Report on Current Folder 则分析当前文件夹下的所有M 文件。

2.Profiler分析工具

Profiler 工具是 MATLAB 提供的另一个功能强大的代码分析工具。在 MATLAB 的Commond Window窗口输入profile viewer命令，出现图2-17 所示的Profiler图形界面。使用时，用户可以提前在Editor中打开M文件，然后运行Profiler分析工具。

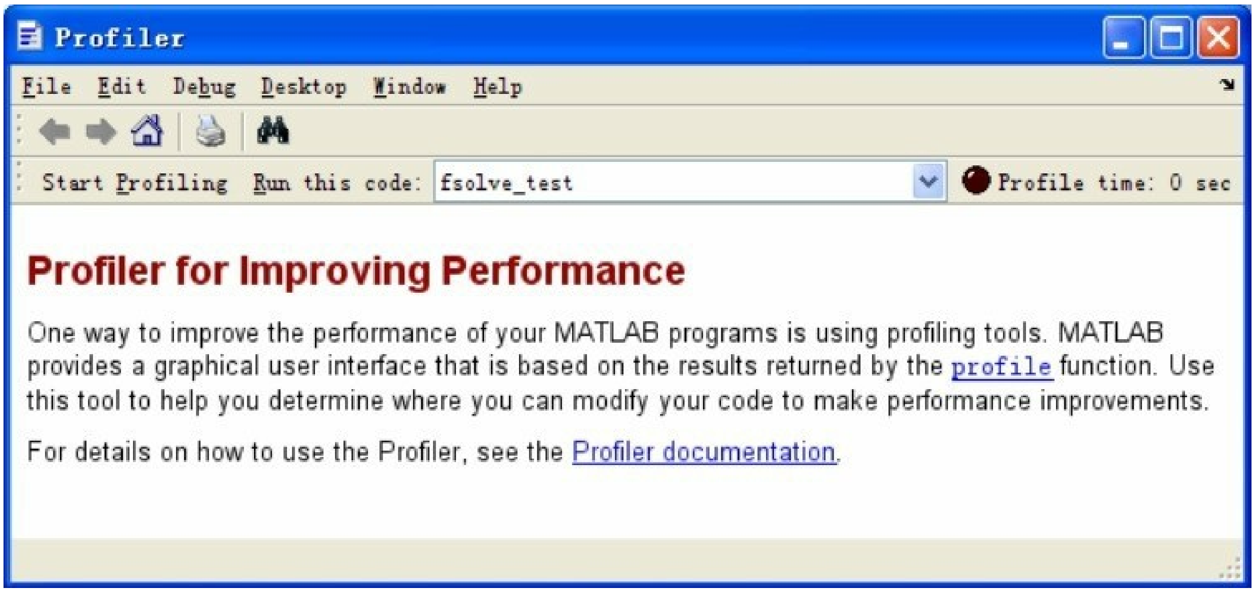


图2-17 Profiler工具图形界面

单击图 2-17 中的“Start Profiling”按钮，就可以分析此 M 文件，分析结果如图 2-18所示。

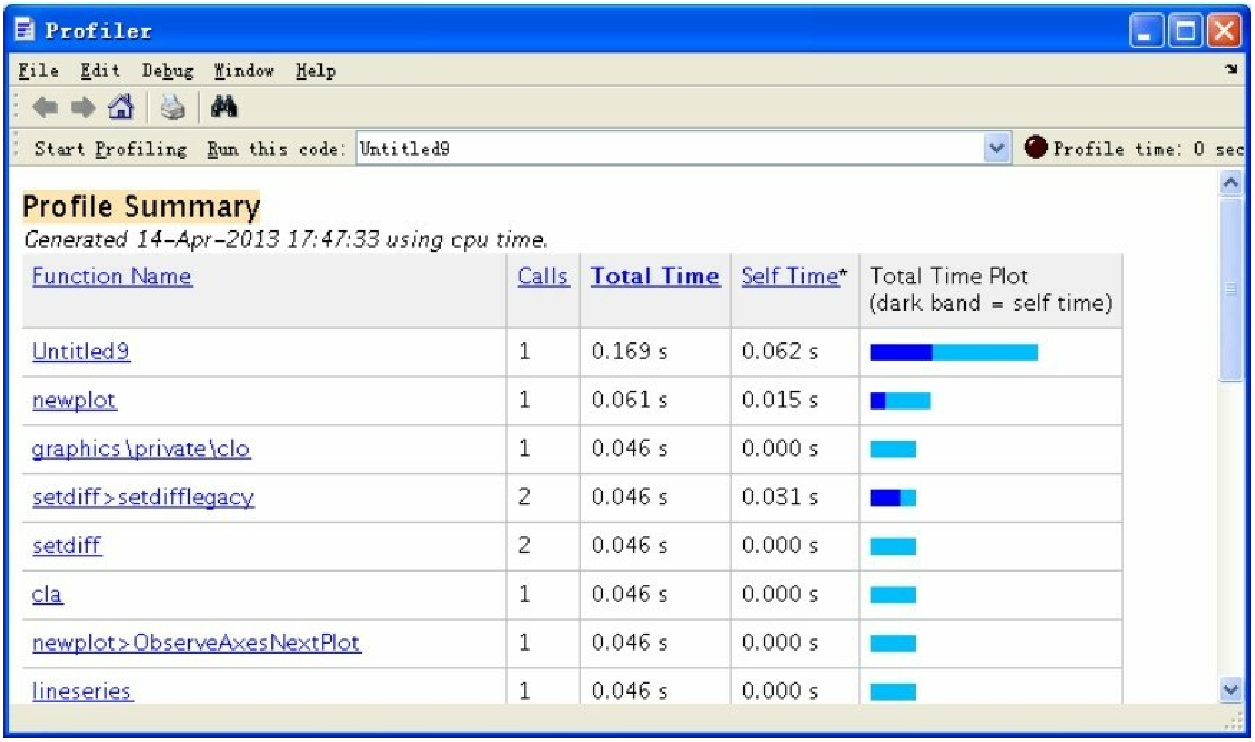


图2-18 Profiler分析结果

从图 2-18 中可见，Profiler 分析结果给出了调用函数名称、调用次数、消耗总时间等信息，单击图2-18 中的“Self Time”，可以使M文件的分析报告按照Self Time 从大到小的方式排列，如图2-19所示。

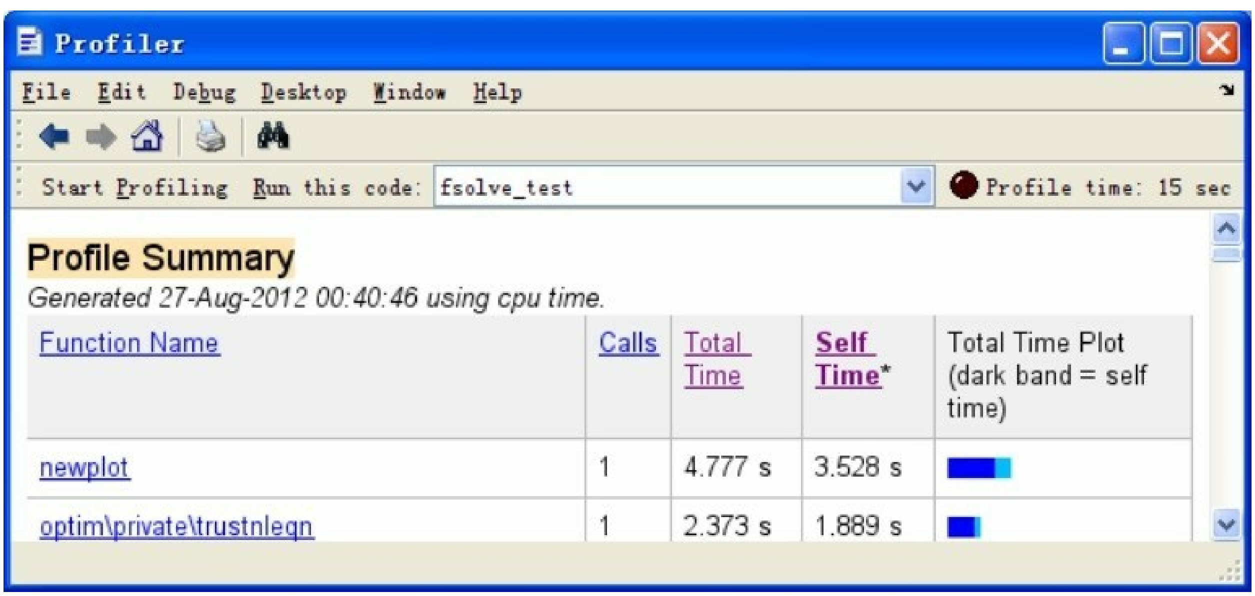


图2-19 文件中耗时顺序排列

图2-19显示了文件运行中最消耗时间的部分及其具体耗时信息。用

户可以有针对性地修改那些最消耗时间的部分。

提示：一般来说，应该尽量避免不必要的变量输出，循环赋值前预定义数组尺寸，多采用向量化的MATLAB函数，少采用数组，这些都能够提高MATLAB程序的运行性能。

2.5 本章小结

数据类型、常量与变量是程序语言入门时必须引入的一些基本概念，MATLAB虽是一个集多种功能于一体的集成软件，但就其语言部分而言，这些概念同样不可缺少。与一般高级语言不同，在MATLAB中数组可作为一个独立的运算单位，直接进行类似简单变量的多种运算而无需采用循环结构，由此决定了数组在MATLAB中作为基本运算量的角色定位。

数据分析和处理在各个领域有着广泛的应用，尤其是在数学、物理等科学领域和工程领域的实际应用中，会经常遇到进行数据分析的情况。

MATLAB跟其他高级计算机语言一样，可以编写MATLAB程序进行程序设计，而与其他计算机语言比起来，有许多无法比拟的优点。本章针对以上内容，均做了详细的介绍。

第3章 MATLAB程序设计

本章介绍 MATLAB 中的 4 大类程序流程控制语句：分支控制语句（if　结构和　switch结构）、循环控制语句（for循环、continue语句和break语句）、错误控制语句和程序终止程序。另外，本章还将介绍MATLAB编程的各种基础知识，同时对于面向对象编程和程序调试也将加以介绍。

学习目标：

·熟练掌握MATLAB 的结构程序

·熟练掌握MATLAB 控制语句

·熟悉面向对象编程方法

·掌握程序调试的方法

3.1 MATLAB的程序结构

MATLAB程序结构一般可分为顺序结构、循环结构、分支结构3种。顺序结构是指按顺序逐条执行，循环结构与分支结构都有其特定的语句，这样可以增强程序的可读性。在这里主要介绍下面4种结构。

3.1.1 if分支结构

如果在程序中需要根据一定条件来执行不同的操作，可以使用条件语句，在MATLAB中提供if分支结构，或者称为if-else-end语句。

根据不同的条件情况，if 分支结构有多种形式，其中最简单的用法是：如果条件表达式为真，则执行语句1，否则跳过该组命令。

if语法结构如下。

```
if 表达式1
   语句1
   else if 表达式2 （可选）
      语句2
   else （可选）
      语句3
   end
end
```

说明如下。

·if 结构是一个条件分支语句，若满足表达式的条件，则往下执行；若不满足，则跳出if结构。

·else if 表达式2 与else 为可选项，这两条语句可依据具体情况取

舍。

注意：每一个if都对应一个end，即有几个if，就应有几个end。

【例3-1】思考下列程序及其运行结果，说明原因。

解：在MATLAB中输入以下程序。

```
>> clear
a=100;
b=20;
if a<b
   fprintf ('b>a')       % 在Word中输入'b>a'单引号不可用，要在Editor中输入
else
   fprintf ('a>b')       % 在Word中输入'b>a'单引号不可用，要在Editor中输入
end
%end
```

运行结果如下。

```
a>b
```

程序说明如下。

程序中，用到了if…else…end的结构，如果a<b，则输出b>a；反之，输出a>b。

由于a=100，b=20，比较可得结果a>b。

在分支结构中，多条语句可以放在同一行，但语句间要用"，"或"；"分开。

【例3-2】思考下列程序及其运行结果，说明原因。

解：在MATLAB中输入以下程序。

```
>> clear
a=20;
```

```
b=20;
if a<b           %   if分支结构
   fprintf('b>a')
else if a==b       %关系符号等于不能写成=,一定要写成==
   fprintf('a=b')
else
   fprintf('a>b')
end
end
%end
```

运行结果如下。

a=b

程序说明：作为等于的关系符一定要写成==。

最后，在使用if分支结构时，需要注意下面的两个问题。

·if 分支结构是所有程序结构中最灵活的结构之一，可以使用任意多个else if语句，但是只能有一个if语句和一个end语句。

·if 语句可以相互嵌套，可以根据实际需要将各个if 语句进行嵌套，从而解决比较复杂的实际问题。

3.1.2 switch分支结构

和C语言中的switch分支结构类似，在MATLAB中适用于条件多而且比较单一的情况，类似于一个数控的多个开关。其一般的语法调用方式如下。

```
switch 表达式
case常量表达式1
   语句组1
```

case常量表达式2

语句组2

………

otherwise

语句组n

end

switch后面的表达式可以是任何类型，如数字、字符串等。

当表达式的值与case后面常量表达式的值相等时，就执行这个case后面的语句组，如果所有的常量表达式的值都与这个表达式的值不相等，则执行otherwise后的语句组。

表达式的值可以重复，在语法上并不错误，但是在执行时，后面符合条件的case语句将被忽略。各个case和otherwise语句的顺序可以互换。

【例3-3】输入一个数，判断它能否被5整除。

解：在MATLAB中输入以下程序。

```
>> clear
n=input('输入n=');        % 输入n值
switchmod(n,5)        %mod是求余函数，余数为0，得0，余数不为0，得1
case 0
   fprintf ('%d是5的倍数',n)
otherwise
   fprintf('%d不是5的倍数',n)
end
%end
```

运行结果如下。

输入n=12

说明12不是5的倍数

在 swith 分支结构中，case 命令后的检测不仅可以为一个标量或者字符串，还可以为一个元胞数组。如果检测值是一个元胞数组，MATLAB将把表达式的值和该元胞数组中的所有元素进行比较；如果元胞数组中某个元素和表达式的值相等，MATLAB认为比较结构为真。

3.1.3 while循环结构

除了上面介绍的分支结构之外，MATLAB还提供多个循环结构。和其他编程语言类似，循环语句一般用于有规律的重复计算。被重复执行的语句称为循环体，控制循环语句流程的语句称为循环条件。

在MATLAB中，while循环结构的语法形式如下。

```
while 逻辑表达式
   循环语句
end
```

while结构依据逻辑表达式的值判断是否执行循环体语句。若表达式的值为真，则执行循环体语句一次，在反复执行时，每次都要进行判断。若表达式为假，则程序执行 end 之后的语句。

为了避免因逻辑上的失误，而陷入死循环，建议在循环体语句的适当位置加 break 语句，以便程序能正常执行。

while 循环也可以嵌套，其结构如下。

```
while 逻辑表达式1
   循环体语句1
while逻辑表达式2
   循环体语句2
end
```

循环体语句3
end
【例3-4】请设计一段程序，求1～100的偶数和。
解：在MATLAB命令窗口输入以下程序。

```
>> clear
x=0;           % 初始化变量x
sum=0;          % 初始化 sum变量
   while x<101       % 当x<101执行循环体语句
   sum=sum+x;   %进行累加
   x=x+2;
   end           % while结构的终点
sum           % 显示 sum
```

运行结果如下。

```
sum =
   2550
```

【例3-5】请设计一段程序，求1～100的奇数和。
解：在MATLAB命令窗口输入以下程序。

```
>> clear
x=1;           % 初始化变量x
sum=0;          % 初始化 sum变量
   while x<101       % 当x<101执行循环体语句
   sum=sum+x;   %进行累加
   x=x+2;
   end           % while结构的终点
sum           % 显示 sum
```

运行结果如下。

```
sum =
```

2500

3.1.4 for循环结构

在MATLAB中，另外一种常见的循环结构是for循环，其常用于知道循环次数的情况，其语法规则如下。

```
for ii=初值：增量：终值
  语句1
  ………
  语句n
end
```

ii=初值：终值，则增量为 1。初值、增量、终值可正可负，可以是整数，也可以是小数，只须符合数学逻辑。

【例3-6】请设计一段程序，求1+2+…+100的和。

解：程序设计如下。

```
>> clear
sum=0;              % 设置初值(必须要有)
for ii=1:100;          %   for循环,增量为1
  sum=sum+ii;
end
sum
%end               % 程序结束
```

运行结果如下。

```
sum =
  5050
```

【例3-7】请说明下列程序及其运行结果，说明原因何在。

解：程序设计如下。

```
>> for ii=1:100;            % for循环,增量为1
   sum=sum+ii;
end
sum
%end                % 程序结束
```

运行结果：1

```
sum =
   10100
```

程序设计：2

```
>> clear
for ii=1:100;            %   for循环,增量为1
sum=sum+ii;
end
sum
%end                % 程序结束
```

运行结果如下。

```
??? Error:"sum" was previously used as a function,
conflicting with its use here as the name of a variable.
```

一般的高级语言中，变量若没有设置初值，程序会以 0 作为其初始值，然而这在MATLAB中是不允许的。所以，在MATLAB中，应给出变量的初值。

程序1没有clear，则程序可能会调用到内存中已经存在的sum值，其结果就成了sum =10100。

在程序2中与上一题的差别是少了sum=0，出现这种情况时，因为程序中有clear语句，则出现错误信息。

【例3-8】请说明下列程序及其运行结果，说明原因何在。

解：在MATLAB中程序设计如下。

```
>> clear
for ii=1:10;
x(ii)=ii.^2;
end
x
%end
```

运行结果如下。

```
x =
   1    4    9    16    25    36    49    64    81    100
```

MATLAB 的变量是以矩阵为基本元素的，这与其他语言是不同的。x 代表一个 1×10的矩阵所以结果是行矩阵而不是x=100。

while循环和for循环都是比较常见的循环结构，但是两个循环结构还是有区别的。其中最明显的区别在于，while循环的执行次数是不确定的，而for循环的执行次数是确定的。

3.2 MATLAB的控制语句

在使用MATLAB设计程序时，经常遇到提前终止循环、跳出子程序、显示错误等情况，因此需要其他的控制语句来实现上面的功能。在MATLAB中，对应的控制语句有continue、break、return等。

3.2.1 continue命令

continue语句通常用于for或while循环体中，其作用就是终止一趟的执行，也就是说它可以跳过本趟循环中未被执行的语句，去执行下一轮的循环。下面使用一个简单的实例，说明continue命令的使用方法。

【例3-9】请思考下列程序及其运行结果，说明原因。

解：在MATLAB中输入以下程序。

```
>> clear
a=3;
b=6;
for ii=1:3
   b=b+1
   if ii<2
   continue
   end          % if语句结束
   a=a+2
end          % for循环结束
%end
```

运行结果如下。

```
b =
    7
b =
    8
a =
    5
b =
    9
a =
    7
```

当 if 条件满足时，程序将不再执行continue后面的语句，而是开始下一轮的循环。continue语句常用于循环体中，与if一同使用。

3.2.2 break命令

break语句也通常用于for或while循环体中，与if一同使用。当if后的表达式为真时就调用break语句，跳出当前的循环。它只终止最内层的循环。

【例3-10】使用break命令跳出当前程序循环。

解：在MATLAB中输入以下程序。

```
>> clear
a=3;
b=6;
for ii=1:3
    b=b+1
    if ii>2
        break
```

```
    end
    a=a+2
end
```

运行结果如下。

```
b =
    7
a =
    5
b =
    8
a =
    7
b =
    9
```

程序说明如下。

·当if 表达式的值为假时，程序执行a=a+2。

·当if 表达式的值为真时，程序执行break 语句，跳出循环。

3.2.3 return命令

在通常情况下，当被调用函数执行完毕后，MATLAB会自动地把控制转至主调函数或者指定窗口。如果在被调函数中插入return命令，可以强制MATLAB结束执行该函数并把控制转出。

return　命令是终止当前命令的执行，并且立即返回到上一级调用函数或等待键盘输入命令，可以用来提前结束程序的运行。

在MATLAB的内置函数中，很多函数的程序代码中引入了return命令，下面引用一个简要的det函数代码，如下。

```
function d=det(A)
if isempty(A)
   a=1;
   return
else
   ...
end
```

在上面的程序代码中，首先通过函数语句来判断函数A的类型，当A是空数组时，直接返回a=1，然后结束程序代码。

3.2.4 input命令

在MATLAB中，input命令的功能是将MATLAB的控制权暂时借给用户，然后，用户通过键盘输入数值、字符串或者表达式，通过按Enter键将输入的内容输入到工作空间中，同时将控制权交给MATLAB，其常用的调用格式如下。

user_entry=input(‘prompt’) 将用户输入的内容赋给变量user_entry。

user_entry=input(‘prompt’,’s’)　将用户输入的内容作为字符串赋给变量user_entry。

【例3-11】在MATLAB中演示如何使用input函数。

解：在MATLAB命令窗口输入如下程序。

```
>>   a=input('input a number:')   %输入数值给 a
input a number:45
a =
   45
b=input('input a number:','s')   %输入字符串给b
input a number:45
```

```
b =
45
input('input a number:')        %将输入值进行运算
input a number:2+3
ans =
   5
```

3.2.5 keyboard命令

在MATLAB中，将keyboard命令放置到M文件中，将使程序暂停运行，等待键盘命令。通过提示符k来显示一种特殊状态，只有当用户使用return命令结束输入后，控制权才交还给程序。在M文件中使用该命令，对程序的调试和在程序运行中修改变量都会十分方便。

【例3-12】在MATLAB中，演示如何使用keyboard命令。

解：在MATLAB命令窗口中输入下面的内容。

```
>> keyboard
K>> for i=1:9
   if i==3
      continue
   end
   fprintf('i=%d\n',i)
   if i==5
      break
   end
end
i=1
i=2
```

```
i=4
i=5
K>> return
>>
```

从上面的程序代码中可以看出，当输入keyboard命令后，在提示符的前面会显示k提示符，而当用户输入return后，提示符恢复正常的提示效果。

在MATLAB中，keyboard命令和input命令的不同在于，keyboard命令运行用户输入任意多个MATLAB命令，而input命令则只能输入赋值给变量的数值。

3.2.6 error和warning命令

在MATLAB中，编写M文件的时候，经常需要提示一些警告信息。为此，MATLAB提供了下面几个常见的命令。

·error('message')：显示出错信息message，终止程序。

·errordlg('errorstring', 'dlgname')：显示出错信息的对话框，对话框的标题为dlgname。

·warning('message')：显示出错信息message，程序继续进行。

【例3-13】查看MATLAB的不同错误提示模式。

解：在MATLAB编辑器中输入以下程序，并将其保存为“error”文件。

```
n=input('Enter:');
if n<2
   error('message');
else
   n=2;
```

```
end
```

返回MATLAB命令窗口，在命令窗口输入“error”，然后分别输入数值1和2，得到如下结果。

```
>> error
Enter:1
Attempt to execute SCRIPT error as a function:
C:\Program Files\MATLAB\R2012a\work\8\error.m
Error in error (line 4)
   error('message');
>> error
Enter:2
```

将上述编辑器中程序修改为如下程序。

```
n=input('Enter:');
if n<2
%       errordlg('Not enough input data','DataError');
   warning('message');
else
   n=2;
end
```

返回MATLAB命令窗口，在命令窗口输入“error”，然后分别输入数值1和2，得到如下结果。

```
>> error
Enter:1
Warning: message
> In error at 4
>> error
Enter:2
```

在上面的程序代码中，演示了MATLAB中不同的错误信息方式。其中，error和warning的主要区别在于warning命令指示警告信息后继续运行程序。

3.3 数据的输入输出

在程序设计中，免不了进行数据的输入与输出，以及与其他外部程序进行数据交换。下面对MATLAB常用的数据输入与输出方法进行介绍。

3.3.1 键盘输入语句（input）

其调用格式如下。

（1）x = input('prompt')：显示提示字符串'prompt'，要求用户键盘输入x的值。

（2）x = input('prompt','s')：显示提示字符串'prompt'，要求用户键盘输入字符型变量x 的值，不至于将输入的数字看成是数值型数据。

3.3.2 屏幕输出语句（disp）

屏幕输出最简单的方法是直接写出欲输出的变量或数组名，后面不加分号。此外，可以采用disp语句，其调用格式为disp(x)。

3.3.3 M数据文件的存储/加载（save/load）

1.save语句

其调用格式如下。

save：将所有工作空间变量存储在名为MATLAB.mat的文件中。

save filename：将所有工作空间变量存储在名为filename 的文件中。

save filename X Y Z：将工作空间的指定变量X、Y、Z 存于名为

filename 的文件中。

2.load语句

其调用格式如下。

load：如果 MATLAB.mat 文件存在，则加载 MATLAB.mat 文件中存储的所有变量到工作空间；否则返回一错误信息。

load filename：如果 filename 文件存在，则加载 filename 文件中存储的所有变量到工作空间；否则返回一错误信息。

load filename X Y Z：如果filename 文件及存储的变量X、Y、Z 存在，则加载filename文件中存储的变量X、Y、Z到工作空间；否则返回一错误信息。

3.3.4 格式化文本文件的存储/读取（fprintf/fscanf）

1.fprintf语句

其调用格式为count = fprintf(fid,format,A,...)，它将用format 定义的格式化文本文件写入以fopen打开的文件（打开文件标识符为文件句柄fid），返回值count为写入文件的字节数。

2.fscanf 语句

其调用格式如下。

A = fscanf(fid,format)：读取以fid 指定的文件数据，并将它转换为format 定义的格式化文本，然后赋给变量A。

[A,count] = fscanf(fid,format,size)：读取以fid 指定的文件数据，读取的数据限定为size 字节，并将它转换为format 定义的格式化文本，然后赋给变量A；同时返回有效读取数据的字节数count。

3.3.5 二进制数据文件的存储/读取（fwrite/fread）

1.fwrite语句

其调用格式为count=fwrite(fid,A,precision)，它将用precision指定的精度，将数组A的元素写入以fid 指定的文件，返回值count 为写入文件的元素数。

2.fread语句

其调用格式为[A,count] = fread(fid,size,precision)，读取以fid 指定的文件中的数组元素，并转换为precision 指定的精度，赋给数组A。返回值count 为成功读取数组的元素数。

3.3.6 数据文件行存储/读取（fgetl/fgets）

1.fgetl语句

其调用格式为 tline = fgetl(fid)，读取以fid 指定的文件中的下一行数据，不包括回车符。

2.fgets语句

其调用格式如下。

tline = fgets(fid)：读取以fid 指定的文件中的下一行数据，包括回车符。

tline = fgets(fid,nchar)：读取以fid 指定的文件中的下一行数据，最多读取nchar 个字符，如果遇到回车符则不再读取数据。

3.4 MATLAB文件操作

常用的文件操作函数列于表 3-1 中。本节仅对文件打开和关闭命令进行介绍，其他命令请读者自行查阅MATLAB帮助文件或参阅其他书籍。

表3-1 常用的文件操作函数

类　别	函　数	说　明
文件打开和关闭	fopen	打开文件，成功则返回非负值
	fclose	关闭文件，可用参数‘all’关闭所有文件
二进制文件	fread	读文件，可控制读入类型和读入长度
	fwrite	写文件
格式化文本文件	fscanf	读文件，与 C 语言中的 fscanf 相似
	fprintf	写文件，与 C 语言中的 fprintf 相似
	fgetl	读入下一行，忽略回车符
	fgets	读入下一行，保留回车符
文件定位	ferror	查询文件的错误状态
	feof	检验是否到文件结尾
	fseek	移动位置指针
	ftell	返回当前位置指针
	frewind	把位置指针指向文件头
临时文件	tempdir	返回系统存放临时文件的目录
	tempname	返回一个临时文件名

1.fopen语句

其常用格式如下。

（1）fid = fopen(filename)：以只读方式打开名为filename 的二进制文件，如果文件可以正常打开，则获得一个文件句柄号fid；否则fid= −1。

（2）fid = fopen(filename,permission)：以permission 指定的方式打开名为filename 的二进制文件或文本文件，如果文件可以正常打开，则获得一个文件句柄号fid（非0整数）；否则fid =−1。

参数permission的设置见表3-2。

表3-2 参数permission 的设置

permission	功　　能
'r'	以只读方式打开文件，默认值
'w'	以写入方式打开或新建文件，如果是存有数据的文件，则删除其中的数据，从文件的开头写入数据
'a'	以写入方式打开或新建文件，从文件的最后追加数据
'r+'	以读/写方式打开文件
'w+'	以读/写方式打开或新建文件，如果是存有数据的文件，写入时则删除其中的数据，从文件的开头写入数据
'a+'	以读/写方式打开或新建文件，写入时从文件的最后追加数据
'A'	以写入方式打开或新建文件，从文件的最后追加数据。在写入过程中不会自动刷新当前输出缓冲区，是为磁带驱动器的写入设计的参数
'W'	以写入方式打开或新建文件，如果是存有数据的文件，则删除其中的数据，从文件的开头写入数据。在写入过程中不会自动刷新当前输出缓冲区，是为磁带驱动器的写入设计的参数

2.fclose 语句

其调用格式如下。

（1）status=fclose(fid)：关闭句柄号fid指定的文件。如果fid是已经打开的文件句柄号， 成功关闭，status =0；否则status=−1。

（2）status = fclose('all')：关闭所有文件（标准的输入/输出和错误信息文件除外）。成功关闭，status =0；否则status=−1。

【例3-14】编写函数，统计M文件中源代码的行数（注释行和空白行不计算在内）。

```
function y =hans(sfile)
% lenm count the code lines of a M-file,
% not include the comments and blank lines
s=deblank(sfile);        %删除文件名 sfile 中的尾部空格
if length(s)<2|| (length(s)>2&&any(lower(s(end-1:end))～='.m'))
   s=[s,'.m'];        %判断有无扩展名.m，若没有，则加上
end
if exist(s,'file')～=2;
   error([s,' not exist']);
```

```
    return;
end
%判断指定的m 文件是否存在；若不存在，则显示错误信息，并返回
line=fgetl(fid);          %逐行读取文件的数据
if isempty(line)||strncmp(deblanks(line),'%',1);
%判断是否为空白行或注释行
    continue;          %若是空白行或注释行则执行下一次循环
end
    count=count+1;       %记录源代码的行数
end
y=count;
function st=deblanks(s);   %删除字符串中的首尾空格的函数
st=fliplr(deblank(fliplr(deblank(s))));
```

以lenm.m 为例，调用并验证该函数。

```
>> sfile=' hans';
>> y = lenm(sfile)
y =
    17
```

3.5 MATLAB程序优化

MATLAB 程序调试工具只能对 M 文件中的语法错误和运行错误进行定位，但是无法评价该程序的性能。程序的性能包括程序的执行效率，内存使用效率，程序的稳定性、准确性及适应性。

MATLAB提供了一个性能剖析指令profile，使用它可以评价程序的性能指标，获得程序各个环节的耗时分析报告。用户可以依据该分析报告寻找程序运行效率低下的原因，以便修改程序。

MATLAB程序优化主要包括效率优化和内存优化两个部分，下面将分别介绍一些常用的优化方法及建议。

3.5.1 效率优化（时间优化）

在程序编写的起始阶段，用户往往将精力集中在程序的功能实现、程序的结构、准确性和可读性等方面，并没有考虑程序的执行效率问题，而是在程序不能够满足需求或者效率太低的情况下才考虑对程序的性能进行优化。因程序所解决的问题不同，程序的效率优化存在差异，这对编程人员的经验以及对函数的编写和调用有一定的要求，一些通用的程序效率优化建议如下。

依据所处理问题的需要，尽量预分配足够大的数组空间，避免在出现循环结构时增加数组空间，但是也要注意不能太大而产生不需要的数组空间，太多的大数组会影响内存的使用效率。

例如预先声明一个 8 位整型数组 A 时，语句 A＝repmat(int8(0),5000,5000)要比A=int8zeros(5000,5000))快 25 倍左右，且更节省内存。因为前者中的双精度 0 仅需一次转换，然后直接申请8 位

整型内存；而后者不但需要为zeros(5000,5000))申请double 型内存空间，而且还需要对每个元素都执行一次类型转换。以下几点值得注意。

·尽量采用函数文件而不是脚本文件，通常运行函数文件都比脚本文件效率更高。

·尽量避免更改已经定义的变量的数据类型和维数。

·合理使用逻辑运算，防止陷入死循环。

·尽量避免不同类型变量间的相互赋值，必要时可以使用中间变量解决。

·尽量采用实数运算，对于复数运算可以转化为多个实数进行运算。

·尽量将运算转化为矩阵的运算。

·尽量使用 MATLAB 的 load、save 指令而避免使用文件的 I/O 操作函数进行文件操作。

以上建议仅供参考，针对不同的应用场合，用户可以有所取舍。有时为了实现复杂的功能不可能将这些要求全部考虑进去。程序的效率优化通常要结合MATLAB的优越性，由于MATLAB的优势是矩阵运算，所以尽量将其他数值运算转化为矩阵的运算，在MATLAB中处理矩阵运算的效率要比简单四则运算更加高效。

3.5.2 内存优化（空间优化）

内存优化对于一些普通的用户而言可以不用顾及，因为随着计算机的发展，内存容量已经能够满足大多数数学运算的要求，而且MATLAB本身对计算机内存优化提供的操作支持较少，只有遇到超大规模运算时，内存优化才能起到作用。下面给出几个比较常见的内存操作函数。

·whos：查看当前内存使用状况函数。

·clear：删除变量及其内存空间，可以减少程序的中间变量。

·save：将某个变量以mat 数据文件的形式存储到磁盘中。

·load：载入mat 数据到内存空间。

由于内存操作函数在函数运行时使用较少，合理的优化内存操作往往由用户编写程序时养成的习惯和经验决定，一些好的做法如下。

·尽量保证创建变量的集中性，最好在函数开始时创建。

·对于含零元素多的大型矩阵，尽量转化为稀疏矩阵。

·及时清除占用内存很大的临时中间变量。

·尽量少开辟新的内存，而是重用内存。

程序的优化本质上也是算法的优化。如果一个算法描述得比较详细的话，它几乎也就指定了程序的每一步。若算法本身描述得不够详细，在编程时会给某些步骤的实现方式留有较大空间，这样就需要找到尽量好的实现方式以达到程序优化的目的。但一般情况下认为算法是足够详细的。如果一个算法设计得足够优的话，就等于从源头上阻止了程序走向劣质。

算法优化的一般要求是，不仅在形式上尽量做到步骤简化，简单易懂，更重要的是能用最小的时间复杂度和空间复杂度完成所需计算。一个好的算法应该包括巧妙的程序流程设计，灵活的循环过程控制（如及时跳出循环或结束本次循环），较好的搜索方式及正确的搜索对象等，以避免不必要的计算过程。

例如在判断一个整数是否是素数时，可以看它能否被m/2以前的整数整除；而更快的方法是只需看它能否被$\sqrt{m}$以前的整数整除，这体现了算法优化的思想。下面通过具体的例子来体会其中所包含的优化思想。

【例3-15】冒泡排序算法。

解：冒泡排序是一种简单的交换排序。其基本思想是两两比较待排

序记录，如果是逆序，则进行交换，直到这个记录中没有逆序的元素。

该算法的基本操作是逐趟进行比较和交换。第一趟比较将最大记录放在 x[n]的位置。一般地，第i趟从x[1]到x[n−i+1]依次比较相邻的两个记录，将这n−i+1个记录中的最大者放在第n−i+1的位置上，其算法程序如下。

```
function s=BubbleSort(x)
% 冒泡排序,x为待排序数组
n=length(x);
for i=1:n-1          % 最多做n-1趟排序
  flag=0;          %   flag为交换标志。本趟排序开始前，交换标志应
为假
  for j=1:n-i      % 每次从前向后扫描，j从1到 n-i
    if x(j)>x(j+1)    %如果前项大于后项则进行交换
      t=x(j+1);
      x(j+1)=x(j);
      x(j)=t;
      flag=1;    % 若发生了交换，则交换标志置为真
    end
  end
  if (～flag)         %若本趟排序未发生交换，则提前终止程序
    break;
  end
end
s=x;
```

说明：本程序通过使用标志变量flag来标志在每一趟排序中是否发生了交换，若某趟排序中一次交换都没有发生则说明此时数组已经为有序（正序），应提前终止算法（跳出循环）。若不使用这样的标志变量

来控制循环往往会增加不必要的计算量。

【例 3-16】公交线路查询问题。设计一个查询算法，给出一个公交线路网中从起始站s1到终到站s2之间的最佳线路。其中一个最简单的情形就是查找直达线路，假设相邻公交车站的平均行驶时间（包括停站时间）为3分钟，若以时间最少为择优标准，请在此简化条件下完成查找直达线路的算法，并根据附录数据（见题后数据1），利用此算法求出以下起始站到终到站之间的最佳路线。

（1）242→105 （2）117→53 （3）179→201 （4）16→162

解：为了便于MATLAB程序计算，应先将线路信息转化为矩阵形式，导入MATLAB （可先将原始数据经过文本导入Excel）。每条线路可用一个一维数组来表示，且将该线路终止站以后的节点用0来表示，每条线路从上往下顺序排列构成矩阵A。

此算法的核心是线路选择问题，要找最佳线路，应先找到所有的可行线路，然后再以所用的时间为关键字选出用时最少的线路。在寻找可行线路时，可先在每条线路中搜索s1，当找到 s1 则接着在该线路中搜索 s2，若又找到 s2，则该线路为一可行线路，记录该线路及所需时间，并结束对该线路的搜索。

另外，在搜索s1与s2时若遇到0节点，则停止对该数组的遍历。

```
%A为线路信息矩阵,s1,s2分别为起始站和终到站。返回值L为最佳线路,t为所需时间
[m,n]=size(A);
L1=[];t1=[];% L1记录可行线路,t1记录对应线路所需时间
for i=1:m
  for j=1:n
    ifA(i,j)==s1          %若找到 s1,则从下一站点开始寻找 s2
      for k=j+1:n
        ifA(i,k)==0      %若此节点为0,则跳出循环
```

```
            break;
          elseifA(i,k)==s2     %若找到   s2,记录该线路及所需时间,
      然后跳出循环
            L1=[L1,i];
            t1=[t1,(k-j)*3];
            break;
          end
        end
      end
    end
  end
  m1=length(L1);         %测可行线路的个数
  ifm1==0             %若没有可行线路，则返回相应信息
    L='No direct line';
    t='Null';
  elseif m1==1
    L=L1;t=t1;          %否则,存在可行线路，用L存放最优线路，t存
放最小的时间
  else
    L=L1(1);t=t1(1);       %分别给L和  t赋初值为第一条可行线路和所
需时间
    for i=2:m1
      if t1(i)< t          %若第 i条可行线路的时间小于 t,
        L=i;       %则给L和 t重新赋值,
        t=t1(i);
      elseif t1(i)==t      %若第 i条可行线路的时间等于 t,
        L=[L,L1(i)];   %则将此线路并入L
```

```
        end
    end
end
```

首先说明，这个程序能正常运行并得到正确结果，但仔细观察之后就会发现它的不足之处。首先是在对j的循环中应先判断节点是否为0，若为0则停止向后访问，转向下一条路的搜索。其次，对于一个二维的数组矩阵，用两层（不是两个）循环进行嵌套就可以遍历整个矩阵，从而得到所有需要的信息。而上面的程序中却出现了三层循环嵌套的局面。

在这种情况下，倘若找到了s2本该停止对此线路节点的访问，但这里的break只能跳出对k的循环，而对该线路数组节点的访问（即对j的循环）将会一直进行到n，做了大量的无用功。

为了消除第三层的循环，能否对第二个循环内的判断语句做如下修改。

```
if A(i,j)==s1
    continue;
    if A(i,k)==s2
        L1=[L1,i];
        t1=[t1,(k-j)*3];
        break;
    end
end
```

这种做法企图控制流程在搜到s1时能继续搜索s2，而不用再嵌套循环。但这样却是行不通的，因为即使s1的后面有s2，也会先被ifA(i,j)==s1拦截，continue后的语句将不被执行，所以经过这番修改后得到的其实是一个错误的程序。

事实上，若想消除第三层循环，可将这第三层循环提出来放在第二

层成为与 j 并列的循环。若在对j的循环中找到了s1，可用一个标志变量对其进行标志，然后再对s1后的节点进行访问，查找s2。综上，可将第一个for循环内的语句修改如下。

```
flag=0;          % 用 flag标志是否找到 s1，为其赋初值为假
for j=1:n
  ifA(i,j)==0      %若该节点为0，则停止对该线路的搜索，转向下一条线路
    break;
  elseifA(i,j)==s1      %否则,若找到 s1，置 flag为真，并跳出循环
    flag=1;
    break;
  end
end
if  flag              %若  flag为真，则找到  s1，从  s1的下一节点开始搜索 s2
  for k=j+1:n
    if A(i,k)==0
      break;
    elseifA(i,k)==s2    %若找到   s2，记录该线路及所需时间，然后跳出循环
      L1=[L1,i];
      t1=[t1,(k-j)*3];
      break;
    end
  end
end
```

若将程序中重叠的部分合并还可以得到一种形式上更简洁的方案。

```
q=s1;                 %用q保存 s1的原始值
for i=1:m
   s1=q;              %每一次给 s1赋初值
   p=0;              %用p值标记是否搜到 s1或 s2
   k=0;              %用k记录站点差
for j=1:n
   if ～A(i,j)
      break;
   elseifA(i,j)==s1       %若搜到  s1,之后在该线路上搜索  s2，并记  p
为1
      p=p+1;
      if p==1
         k=j-k;
         s1=s2;
      elseifp==2       %当p值为 2时，说明已搜到 s2，记录相关信息
         L1=[L1,i];
            t1=[t1,3*k];   %同时  s1恢复至原始值，进行下一线路的搜
      索
            break;
         end
      end
   end
end
```

程序运行如下。

```
[L,t]=DirectLineSearch(242,105,A)
L =
   8
```

t =

24

?[L,t]=DirectLineSearch(117,53,A)

L =

10

t =

15

[L,t]=DirectLineSearch(179,201,A)

L =

7 14

t =

27

?[L,t]=DirectLineSearch(16,162,A)

L =

No direct line

t =

Null

注意：在设计算法或循环控制时，应注意信息获取的途径，避免做无用的操作步骤。如果上面这个程序不够优化，它将为后续转车的程序造成不良影响。

附录数据1：公交线路信息

线路1

219-114-88-48-392-29-36-16-312-19-324-20-314-128-76-113-110-213-14-301-115-34-251-95-184-92

线路2

348-160-223-44-237-147-201-219-321-138-83-161-66-129-254-331-317-303-127-68

线路3

23-133-213-236-12-168-47-198-12-236-113-212-233-18-127-303-117-231-254-129-366-16 1-133-181-132

线路4

201-207-177-144-223-216-48-42-280-140-238-236-158-53-93-64-130-77-264-208-286-123

线路5

217-272-173-25-33-76-37-27-65-274-234-221-137-306-162-84-325-97-89-24

线路6

301-82-79-94-41-105-142-118-130-36-252-172-57-20-302-65-32-24-92-218-31

线路7

184-31-69-179-84-212-99-224-232-157-68-54-201-57-172-22-36-143-218-129-106-101-194

线路8

57-52-31-242-18-353-33-60-43-41-246-105-28-33-111-77-49-67-27-8-63-39-317-168-12-163

线路9

217-161-311-25-29-19-171-45-71-173-129-219-210-35-83-43-139-241-78-50

线路10

136-208-23-117-77-130-68-45-53-51-78-241-139-343-83-333-190-237-251-291-129-173-1 71-90-42-179-25-311-161-17

线路11

43-77-111-303-28-65-246-99-54-37-303-53-18-242-195-236-26-40-280-142

线路12

274-302-151-297-329-123-122-215-218-102-293-86-15-215-186-213-105-128-201-122-12-29-56-79-141-24-74

线路13

135-74-16-108-58-274-53-59-43-86-85-47-246-108-199-296-261-203-227-146

线路14

224-22-70-89-219-228-326-179-49-154-251-262-307-294-208-24-201-261-192-264-146-37 7-172-123-61-235-294-28-94-57-226-18

线路15

189-170-222-24-92-184-254-215-345-315-301-214-213-210-113-263-12-167-177-313-219-154-349-316-44-52-19

线路16

233-377-327-97-46-227-203-261-276-199-108-246-227-45-346-243-59-93-274-58-118-116-74-135

事实上，对于编程能力的训练，往往是先从解决一些较为简单问题入手，然后通过对这些问题修改某些条件，增加难度等不断地进行摸索，在不知不觉中自己的编程能力就已经被提升到了一个新的高度。

3.5.3 编程注意事项

1.有关程序的拆分与组合

在编写一个蕴含多个程序模块的较大程序时，通常还要考虑什么时候应将各个程序模块（子程序）分开来写，便于其他函数调用，什么时候又适合用一个函数整体完成流程。

对于一些较为典型的算法，或者某个独立的计算过程会在以后的计算中多次被用到，最好将这样的计算过程写成独立的函数，以便被其他

函数调用，或是被后续的计算过程使用。若是在被其他函数调用时需要对某些参数进行修改，最好将这些参数设置为从函数输入的形式，这就要求在编写函数或函数模块时应尽量考虑其通用性。

另外，在编程过程中能用矩阵操作完成的尽量不用循环，因为在MATLAB中，矩阵语句被直接翻译成逻辑变量（0，1变量）执行，而循环语句则是采用逐行翻译的方式首先翻译成MATLAB的母语句，执行速度较前者会大大降低。

2.其他

对于一个较复杂的程序最好是先写出较详细的算法步骤，然后在算法的指导下进行编程，以免直接进行编程时很多地方考虑不周，以至寸步难行。

若某个变量值只是在程序运行时被显示，而并非作为函数输出值，则这个变量值不能作为其他函数的输入被直接使用，若想使用它需将其包含在函数的输出项中。

要养成写注释的习惯，以增强程序的可读性。

当所要处理的数据在 Excel 表中（或这些数据可以导入 Excel），尤其是当数据量较大时，将Excel表导入MATLAB就显得较为必要。其导入方法如下。

（1）启动Excel。

（2）单击“工具”加载宏命令，在弹出的加载宏对话框中单击“浏览”按钮。

（3）找到MATLAB 7\toolbox\exlink.xla。

（4）返回加载宏对话窗口，单击“确定”按钮。

建立连接后，可利用“putmatriax”与“getmatriax”实现Excel与MATLAB之间的数据交换。

文本数据导入Excel的方法如下。

（1）启动Excel。

（2）单击“数据”导入外部数据\外部数据命令，在弹出的选择数据源对话框中找到文本数据，单击“打开”按钮。

（3）设置分隔符及其他操作。

（4）单击“完成”及“确定”按钮。

3.5.4 几个常用数学方法的算法程序

1.雅可比（Jacobi）迭代算法

该算法是解方程组的一个较常用的迭代算法。

```
function x=ykb(A,b,x0,tol)
% A为系数矩阵，b为右端项，x0(列向量)为迭代初值,tol 为精度
D=diag(diag(A));%将A分解为D,-L,-U
L=-tril(A,-1);
U=-triu(A,1);
B1=D\(L+U);
f1=D\b;
q=norm(B1);
d=1;
while q*d/(1-q)>tol %迭代过程
   x=B1*x0+f1;
   d=norm(x-x0);
   x0=x;
end
```

2.拉格朗日（Lagrange）插值函数算法

该算法用于求解插值点处的函数值。

```
function y=lagr1(x0,y0,x)
% x0，y0为已知点列，x为待插值节点（可为数组）
```

```
% 当输入参数只有x0，y0时，返回y 为插值函数
% 当输入参数有x时返回y 为插值函数在x处所对应的函数值
n=length(x0);
if nargin==2
    syms x
    y=0;
    for i=1:n
        L=1;
        for j=1:n
            if j～=i
                L=L*(x-x0(j))/(x0(i)-x0(j));
            end
        end
        y=y+L*y0(i);
        y=simple(y);
    end
    x1=x0(1):0.01:x0(n);
    y1=subs(y,x1);
    plot(x1,y1);
else
    m=length(x);
    for k=1:m % 对每个插值节点分别求值
        s=0;
            for i=1:n
                L=1;
                for j=1:n
                    if j～=i
```

```
            L=L*(x(k)-x0(j))/(x0(i)-x0(j));
          end
        end
        s=s+L*y0(i);
      end
    end
```

注意：以上两个算法属数值计算类，注意对比其解析表达式与用程序进行数值计算时在操作方式上的不同。

3.图论相关算法

（1）最小生成树。

```
function [w,E]=MinTree(A)
% 避圈法求最小生成树
% A为图的赋权邻接矩阵
% w记录最小树的权值之和，E记录最小树上的边
n=size(A,1);
for i=1:n
A(i,i)=inf;
end
s1=[];s2=[];      % s1、s2记录一条边上的两个顶点
w=0; k=1;      % k记录顶点数
T=A+inf;
T(1,:)=A(1,:);
A(:,1)=inf;
while k<n
  [p1,q1]=min(T);   % q1记录行下标
  [p2,q2]=min(p1);
  i=q1(q2);
```

```
    s1=[s1,i];s2=[s2,q2];
    w=w+p; k=k+1;
    A(:,q2)=inf;    % 若此顶点已被连接，则切断此顶点的入口
    T(q2,:)=A(q2,:);    % 在T中并入此顶点的出口
    T(:,q2)=inf;
end
E=[s1;s2];        %E记录最小树上的边
```

（2）最短路的Dijkstra算法。

```
function [d,path]=ShortPath(A,s,t)
% Dijkstra 最短路算法实现,A为图的赋权邻接矩阵
% 当输入参数含有s 和t时，求s 到t的最短路
% 当输入参数只有s时，求s 到其他顶点的最短路
% 返回值d为最短路权值，path为最短路径
if nargin==2
    flag=0;
elseif nargin==3
    flag=1;
end
n=length(A);
for i=1:n
    A(i,i)=inf;
end
V=zeros(1,n);            % 存储 lamda(由来边)标号值
D=zeros(1,n);            % 用D记录权值
T=A+inf;                 %T为标号矩阵
T(s,:)=A(s,:);           % 先给起点标号
A(:,s)=inf;              % 关闭进入起点的边
```

```
for k=1:n-1
   [p,q]=min(T);          % p记录各列最小值，q为对应的行下标
   q1=q;             % 用q1保留行下标
   [p,q]=min(p);          %求最小权值及其列下标
   V(q)=q1(q);          %求该顶点 lamda值
   if flag&q==t
      d=p;          % 求最短路权值
      break;
   else              % 修改T标号：
      D(q)=p;          % 求最短路权值
      A(:,q)=inf;       % 将A中第q列的值改为 inf
      T(q,:)=A(q,:)+p;       %同时修改从顶点q出去的边上的权值
      T(:,q)=inf;       % 顶点q点已完成标号，将进入q的边关闭
   end
end
if flag              % 输入参数含有 s和 t，求 s到 t的最短路
   path=t;          %逆向搜索路径
   while path(1)～=s
      path=[V(t),path];
      t=V(t);
   end
else                % 输入参数只有 s，求 s到其他顶点的最短路
   for i=1:n
      if i～=s
         path0=i;v0=i;   %逆向搜索路径
         while path0(1)～=s
            path0=[V(i),path0];
```

```
            i=V(i);
        end
        d=D; path(v0)={path0}; %将路径信息存放在元胞数组中
        % 在命令窗口显示权值和路径
        disp([int2str(s),'->',int2str(v0),' d=',...
        int2str(D(v0)),' path= ',int2str(path0)]);
      end
   end
end
```

（3）Ford最短路算法：该算法用于求解从源点逐次绕过其他顶点，以缩短到达终点的最短路径长度。

```
function [w,v]=Ford(W,s,t)
% W为图的带权邻接矩阵,s为发点,t为终点
% 返回值w为最短路的权值之和,v为最短路线上的顶点下标
n=length(W);
d(:,1)=(W(s,:))';            % 求d(vs,vj)=min{d(vs,vi)+wij}的解, 用d存放
% d(t)(v1,vj), 赋初值为W的第s行,以列存放
j=1;
while j
   for i=1:n
      b(i)=min(W(:,i)+d(:,j));
   end
   j=j+1;
   d=[d,b'];
   if d(:,j)==d(:,j-1)       %若找到最短路, 跳出循环
      break ;
   end
```

```
end
w=d(t,j);              % 记录最短路的权值之和
v=t;                   % 用数组v存放最短路上的顶点,终点为 t
while v(1)～=s
   for i=n:-1:1
      if i～=t&W(i,t)+d(i,j)==d(t,j)
         break;
      end
   end
   v=[i,v];
   t=i;
end
```

4.模糊聚类分析算法程序（组）

在模糊聚类分析中，该算法中的“程序_3”用于求解模糊矩阵、模糊相似矩阵和模糊等价矩阵。“程序_4”用来完成聚类。“程序_1”和“程序_2”是为“程序_3”服务的子程序（有关模糊聚类分析的有关知识可查阅相关资料）。

程序_1：求模糊合成矩阵的最大最小法。

```
function s=mhhc(R1,R2)                 % 模糊合成
[m,n]=size(R1);
[n,n1]=size(R2);
for i=1:m
   for j=1:n1
      s(i,j)=max(min(R1(i,:),(R2(:,j))'));       %最大最小法
   end
end
```

注意：此函数被程序_2调用。

程序_2：求模糊传递包的算法。

```
function s=mhcdb(R)
% 求模糊传递包
while sum(sum(R～=mhhc(R,R))) %调用模糊合成函数‘mhhc’
   R=mhhc(R,R);
end
s=R;
```

注意：此函数被程序_3调用。

程序_3。

```
for j=1:n
   s1(j)= sqrt(sum((x(:,j)-x0(j)).^2)/m);    %对x做平移——标准差变换
   x(:,j)=(x(:,j)-x0(j))/s1(j);
   x1(:,j)=(x(:,j)-min(x(:,j)))/(max(x(:,j))-min(x(:,j)));
   % 平移——极差变换
end
s1=x1;                          % s1 表示模糊矩阵
R=eye(m);
M=0;                            % 相似系数 r由数量积法求得
for i=1:m
   for j=i+1:m
      if(sum(x1(i,:).*x1(j,:))>M);
         M=sum(x1(i,:).*x1(j,:));
      end
   end
end
for i=1:m
   for j=1:m
```

```
        if(i～=j)
            R(i,j)=(sum(x1(i,:).*x1(j,:)))/M;
        end
    end
end
s2=R;                    %R为模糊相似矩阵
s3=mhcdb(R);                    %                    s3表示模糊等价矩阵    ,此处调
```

用‘mhcdb’求模糊传递包

注意：本程序中若想用夹角余弦法求相似系数r，可将上面程序中的第14行（M=0；）至第28行（倒数第3行）用下面的程序段替换。

```
for i=1:m % 夹角余弦法求相似系数r
    for j=1:m
        M1=sqrt(sum(x1(i,:).^2)*sum(x1(j,:).^2));
        R(i,j)=(sum(x1(i,:).*x1(j,:)))/M1;
    end
end
```

程序_4。

```
function [L1,s]=Lamjjz(x,lam)
% 求λ-截矩阵并完成聚类,x为模糊等价矩阵
%(即程序_3中求得的s3),lam为待输入的λ值
n=length(x(1,:));
for i=1:n
    for j=1:n
        if x(i,j)>=lam
            L1(i,j)=1;        % x1为λ-截矩阵
        end
    end
```

```
end
A=zeros(n,n+1);
for i=1:n
if ～A(i,1)
A(i,2)=i;                %A的第一列为标示符其值为0或1
for j=i+1:n
   if x1(i,:)==x1(j,:)
      A(i,j+1)=j;
      A(j,1)=1;
   end
end
for i=1:n
   if ～A(i,1)
      a=[];
      for j=2:n+1
         if A(i,j)
            a=[a,A(i,j)];      % a表示聚类数组
         end
      end
      disp(a)              % 将聚类数组依次显示
   end
end
```

5.层次分析——求近似特征向量算法

在层次分析中，该算法用于根据成对比较矩阵求近似特征向量。

```
function [w,lam,CR]=ccfx(A)
% A为成对比较矩阵,返回值w为近似特征向量,
% lam为近似最大特征值maxλ,CR为一致性比率
```

```
n=length(A(:,1));
a=sum(A);
B=A;                        % 用B代替A做计算
for j=1:n                        % 将A的列向量归一化
   B(:,j)=B(:,j)./a(j);
end
s=B(:,1);
for j=2:n
   s=s+B(:,j);
end
c=sum(s);                     % 和法计算近似最大特征值maxλ
w=s./c;
d=A*w;
lam=1/n*sum((d./w));
CI=(lam-n)/(n-1);               % 一致性指标
RI=[0,0,0.58,0.90,1.12,1.24,1.32,1.41,1.45,1.49,1.51];
% RI为随机一致性指标
CR=CI/RI(n);                   %求一致性比率
if CR>0.1
   disp('没有通过一致性检验');
   else disp('通过一致性检验');
end
```

6.灰色关联性分析——单因子情形

当系统的行为特征只有一个因子0x，该算法用于求解各种因素ix对0x的影响大小。

```
function s=Glfx(x0,x) % x0(行向量)为因子,x为因素集
[m,n]=size(x);
```

```
B=[x0;x];
k=m+1;% k为B的行数
c=B(:,1);% 对序列进行无量纲化处理
for j=1:n
   B(:,j)=B(:,j)./c;
end
for i=2:k              % 求参考序列对各比较序列的绝对差
   B(i,:)=abs(B(i,:)-B(1,:));
end
A=B(2:k,:);              %求关联系数
a=min(min(A));
b=max(max(A));
for i=1:m
   for j=1:n
      r1(i,j)=r1(i,j)*(a+0.5*b)/(A(i,j)+0.5*b);
   end
end
s=1/n*(r1*ones(m,1));       % 比较序列对参考序列x0的灰关联度
```

7.灰色预测——GM(1,1)

该算法用灰色模型中的GM(1,1)模型做预测。

```
function [s,t]=huiseyc(x,m)
% x为待预测变量的原值,为其预测m个值
[m1,n]=size(x);
ifm1～=1              % 若x为列向量,则将其变为行向量放入x0
   x0=x';
else
   x0=x;
```

```
end
n=length(x0);
c=min(x0);
if c<0              % 若x0中有小于0的数,则作平移,使每个数字都大于0
   x0=x0-c+1;
end
x1=(cumsum(x0))';          % x1为x0的 1次累加生成序列,即AGO
for k=2:n
   r(k-1)=x0(k)/x1(k-1);
end
rho=r,              % 光滑性检验
for k=2:n
   z1(k-1)=0.5*x1(k)+0.5*x1(k-1);
end
B=[-z1',ones(n-1,1)];
YN=(x0(2:n))';
a=(inv(B'*B))*B'*YN;
y1(1)=x0(1);
for k=2:n+m              % 预测m个值
   y1(k)=(x0(1)-a(2)/a(1))*exp(-a(1)*(k-1))+a(2)/a(1);
end
y(1)=y1(1);
for k=2:n+m
   y(k)=y1(k)-y1(k-1);% 还原
end
if c<0
   y=y+c-1;
```

```
end
y;
e1=x0-y(1:n);
e=e1(2:n),              % e为残差
for k=2:n
   dd(k-1)=abs(e(k-1))/x0(k);
end
dd;
d=1/(n-1)*sum(dd);
f=1/(n-1)*abs(sum(e));
s=y;
t=e;
```

以上程序实例仅供参考练习，要想使自己的编程水平得到根本性的提高，除了学习相关知识和经验外，更重要的是，一定要自己主动去编写程序，可先从编写一些较简单的程序入手，然后逐渐增大难度，多加练习，善于摸索一些特殊问题的处理方法和技巧，或者通过查书、与别人讨论等方式来丰富自己的经验，提升自己的编程能力。

3.6 程序调试

程序调试的目的是检查程序是否正确，即程序能否顺利运行并得到预期结果。在运行程序之前，应先设想到程序运行的各种情况，测试在各种情况下程序是否能正常运行。

对初学编程的人来说，很难保证所编的每个程序都能一次性运行通过，而大多情况下都需要对程序进行反复的调试之后才能正确运行。所以，不要害怕程序出错，要时时准备着去查找错误，改正错误。

3.6.1 程序调试命令

MATLAB提供了一系列程序调试命令，利用这些命令，可以在调试过程中设置、清除和列出断点，逐行运行M文件，在不同的工作区检查变量，用来跟踪和控制程序的运行，帮助寻找和发现错误。所有的程序调试命令都是以字母db开头的，见表3-3。

表3-3 程序调试命令

命　　令	功　　能
dbstop in fname	在 M 文件 fname 的第一可执行程序上设置断点
dbstop at r in fname	在 M 文件 fname 的第 r 行程序上设置断点
dbstop if v	当遇到条件 v 时，停止运行程序。当发生错误时，条件 v 可以是 error，当发生 NaN 或 inf 时，也可以是 naninf/infnan
dstop if warning	如果有警告，则停止运行程序
dbclear at r in fname	清除文件 fname 的第 r 行处断点
dbclear all in fname	清除文件 fname 中的所有断点
dbclear all	清除所有 M 文件中的所有断点
dbclear in fname	清除文件 fname 第一可执行程序上的所有断点
dbclear if v	清除第 v 行由 dbstop if v 设置的断点
dbstatus fname	在文件 fname 中列出所有的断点
Mdbstatus	显示存放在 dbstatus 中用分号隔开的行数信息
dbstep	运行 M 文件的下一行程序
dbstep n	执行下 n 行程序，然后停止
dbstep in	在下一个调用函数的第一可执行程序处停止运行
dbcont	执行所有行程序直至遇到下一个断点或到达文件尾
dbquit	退出调试模式

进行程序调试，要调用带有一个断点的函数。当MATLAB进入调试模式时，提示符为K>>。最重要的区别在于现在能访问函数的局部变量，但不能访问MATLAB工作区中的变量。具体的调试技术，请读者在调试程序的过程中逐渐体会。

3.6.2 程序剖析

对于简单的MATLAB程序中出现的语法错误，可以采用直接调试法，即直接运行该M文件，MATLAB将直接找出语法错误的类型和出现的地方，根据MATLAB的反馈信息对语法错误进行修改。

当M文件很大或M文件中含有复杂的嵌套时，则需要使用MATLAB调试器来对程序进行调试，即使用MATLAB提供的大量调试函数以及与之相对应的图形化工具。

下面通过一个判断2000年至2010年间的闰年年份的示例来介绍MATLAB调试器的使用方法。

【例3-17】编写一个判断2000年至2010年间的闰年年份的程序并调试。

解：（1）创建一个leapyear.m的M函数文件，并输入如下函数代码程序。

```
%程序为判断2000年至2010年10年间的闰年年份
%本程序没有输入/输出变量
%函数的使用格式为leapyear，输出结果为2000年至2010年10年间的闰年年份
function leapyear       %定义函数 leapyear
foryear=2000:2010    %定义循环区间
   sign=1;
   a= rem(year,100);    %求year除以100后的剩余数
   b= rem(year,4);    %求year除以4后的剩余数
   c= rem(year,400);    %求year除以400后的剩余数
   if a=0       %以下根据 a、b、c是否为0对标志变量 sign进行处理
      signsign=sign-1;
   end
   if b=0
      signsign=sign+1;
   end
   if c=0
      signsign=sign+1;
   end
   if sign=1
      fprintf('%4d \n',year)
```

```
    end
end
```

（2）运行以上M程序，此时MATLAB命令窗口会给出如下错误提示。

```
>> leapyear
Error: File: leapyear.m Line: 10 Column: 10
The expression to the left of the equals sign is not a valid target
for an assignment.
```

由错误提示可知，在程序的第10行存在语法错误，检测可知if选择判断语句中，用户将“==”写成了“=”。因此将“=”改成“==”，同时也更改第13、16、19行中的“=”为“==”。

（3）程序修改并保存完成后，可直接运行修正后的程序，程序运行结果如下。

```
>> leapyear
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
```

显然，2001年至2010年间不可能每年都是闰年，由此判断程序存在运行错误。

（4）分析原因。可能由于在处理年号是否是100的倍数时，变量sign存在逻辑错误。

（5）断点设置。断点为MATLAB程序执行时人为设置的中断点，程序运行至断点时便自动停止运行，等待用户的下一步操作。设置断点只需要用鼠标单击程序左侧的“-”使得“-”变成红色的圆点（当存在语法错误时圆点颜色为灰色），如图3-1所示。

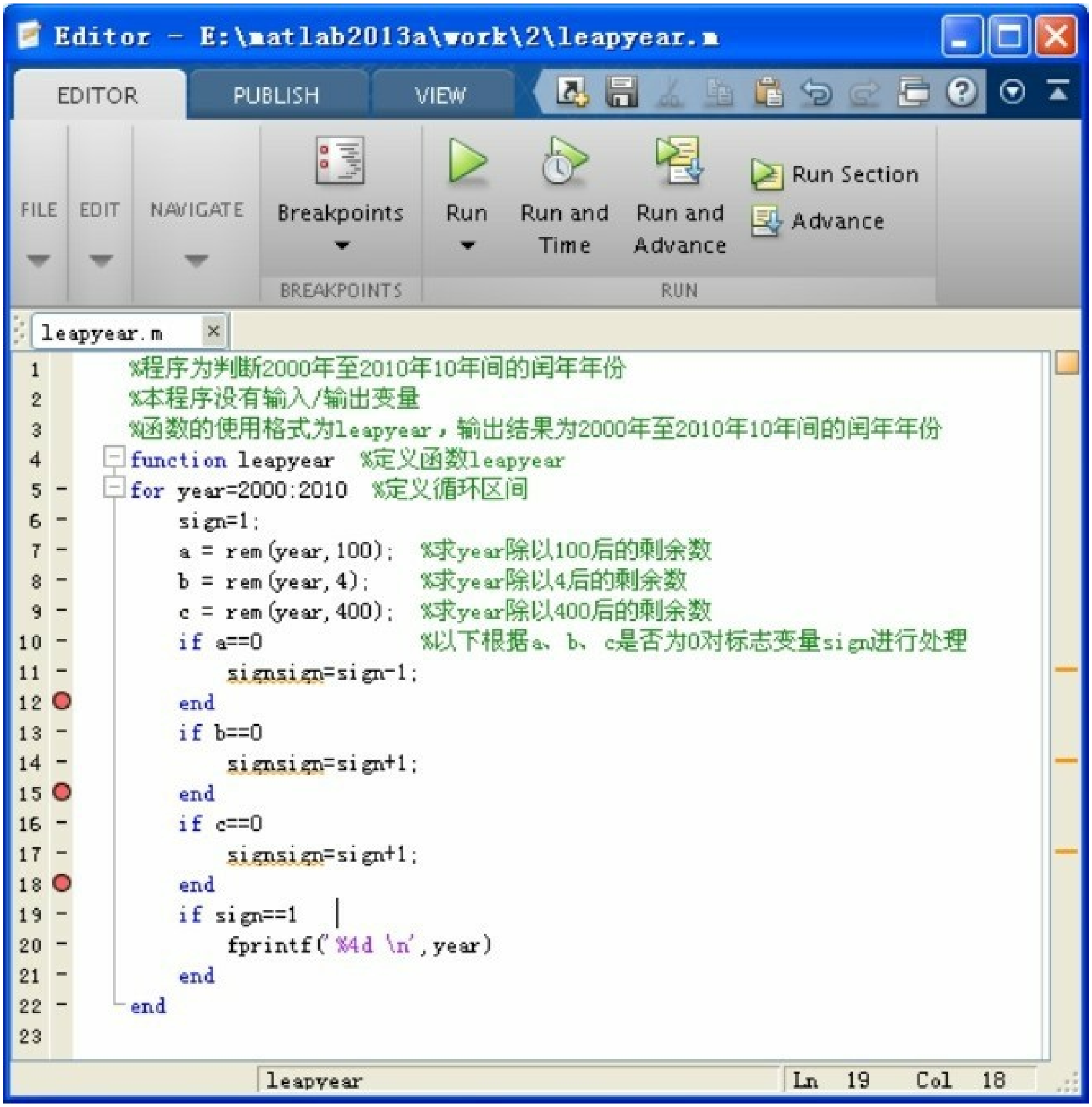
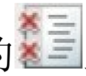


图3-1 断点标记

应该在可能存在逻辑错误或需要显示相关代码执行数据附近设置断点，例如，本例中的第12行、第15行和第18行。如果用户需要去除断点，可以再次单击红色圆点去除，也可以单击工具栏中的工具去除所有断点。

（6）运行程序。按F5键或单击工具栏中的按钮执行程序，这时其他调试按钮将被激活。程序运行至第一个断点暂停，在断点右侧则出现向右指向的绿色箭头，如图3-2所示。

进行查看。

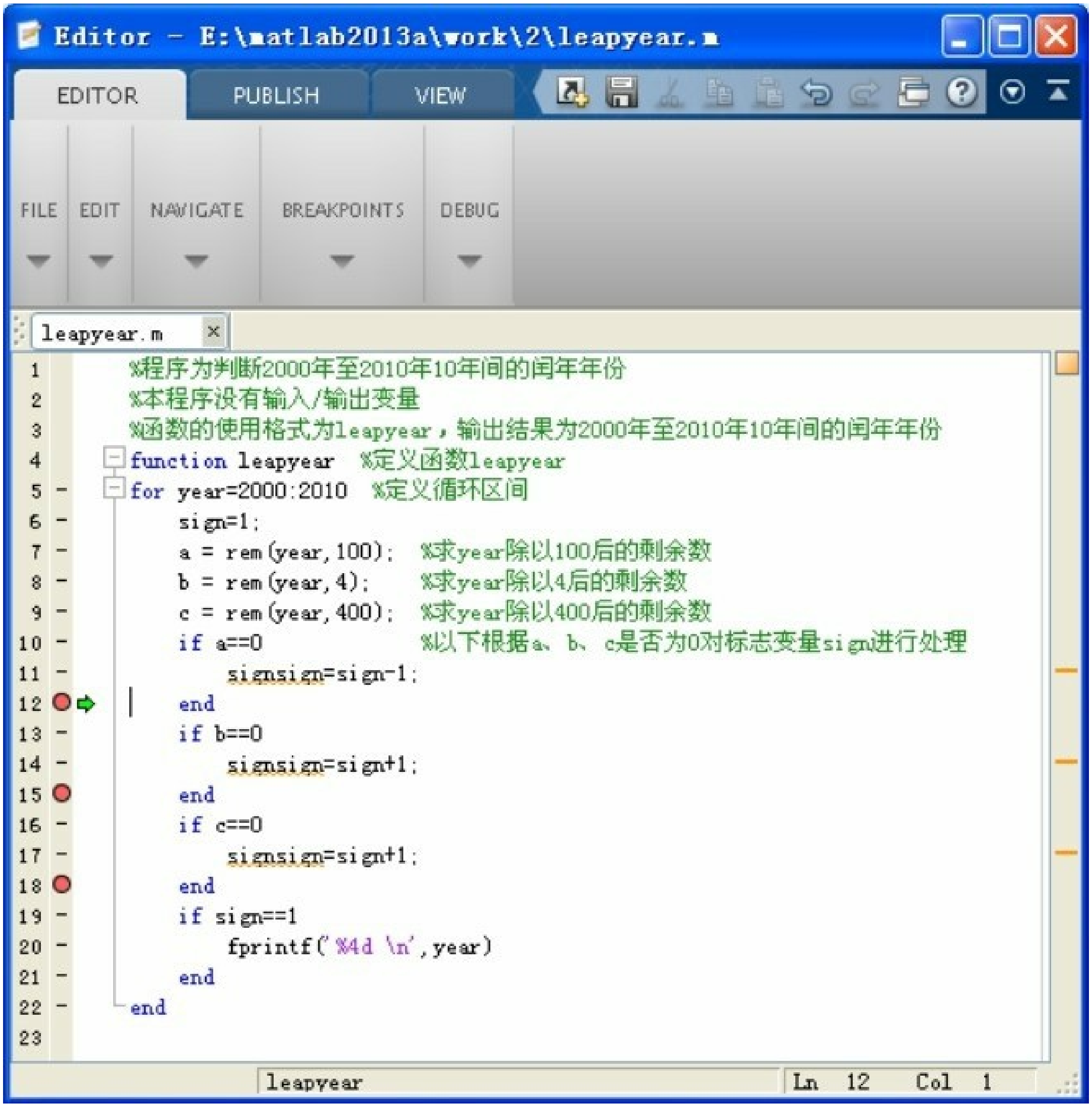


图3-2 程序运行至断点处暂停

程序调试运行时，在MATLAB的命令窗口中将显示如下内容。

```
>> leapyear
12      end
K>>
```

此时可以输入一些调试指令，更加方便对程序调试的相关中间变量进行查看。

（7）单步调试。可以通过按F10键或单击工具栏中相应的单步执行图形按钮，此时程序将一步一步按照用户需求向下执行，如图3-3所示，在单击F10键后，程序从第12步运行到第13步。

（8）查看中间变量。可以将鼠标停留在某个变量上，MATLAB将会自动显示该变量的当前值，也可以在MATLAB的workspace中直接查看所有中间变量的当前值，如图3-4和图3-5所示。

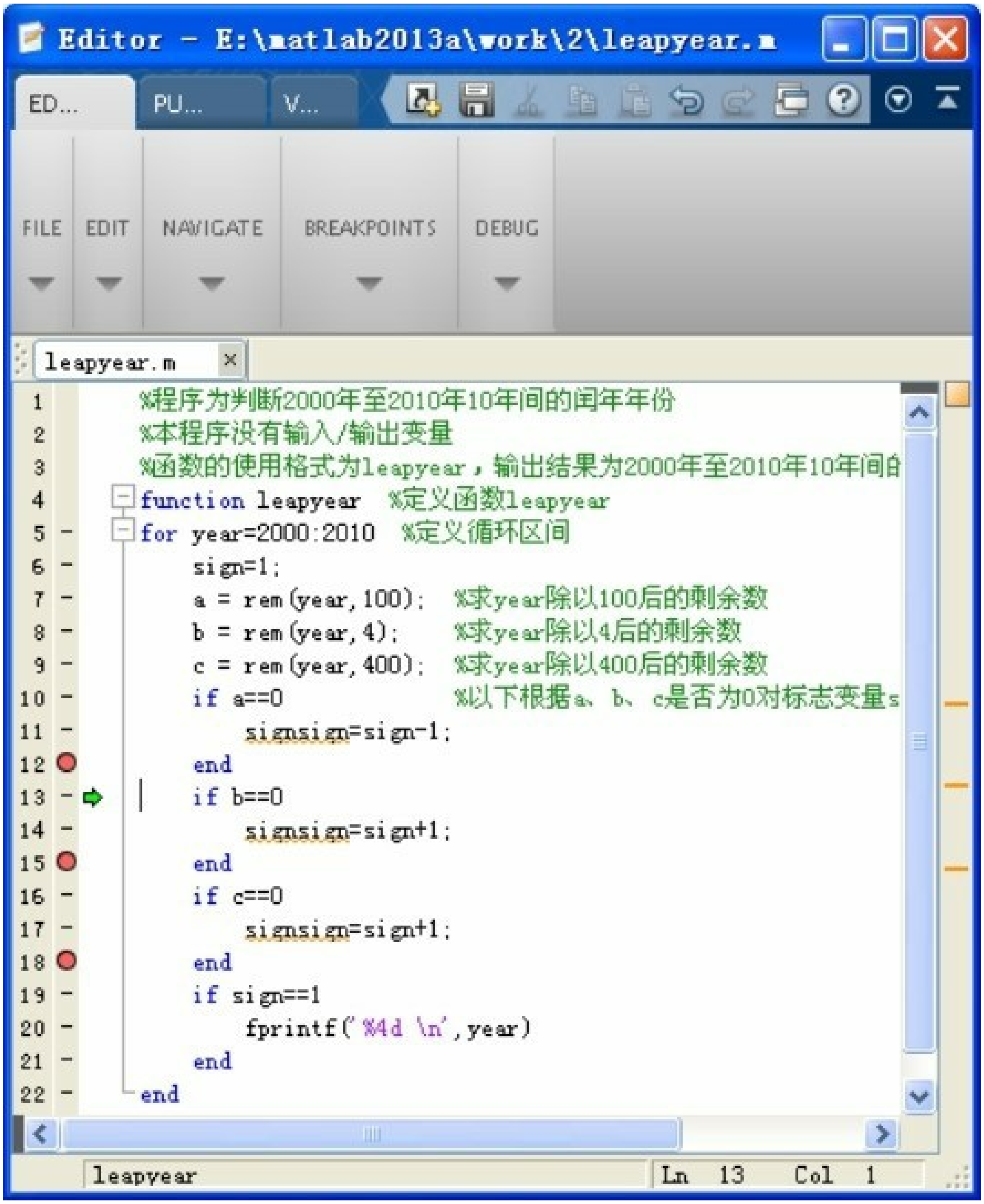


图3-3 程序单步执行

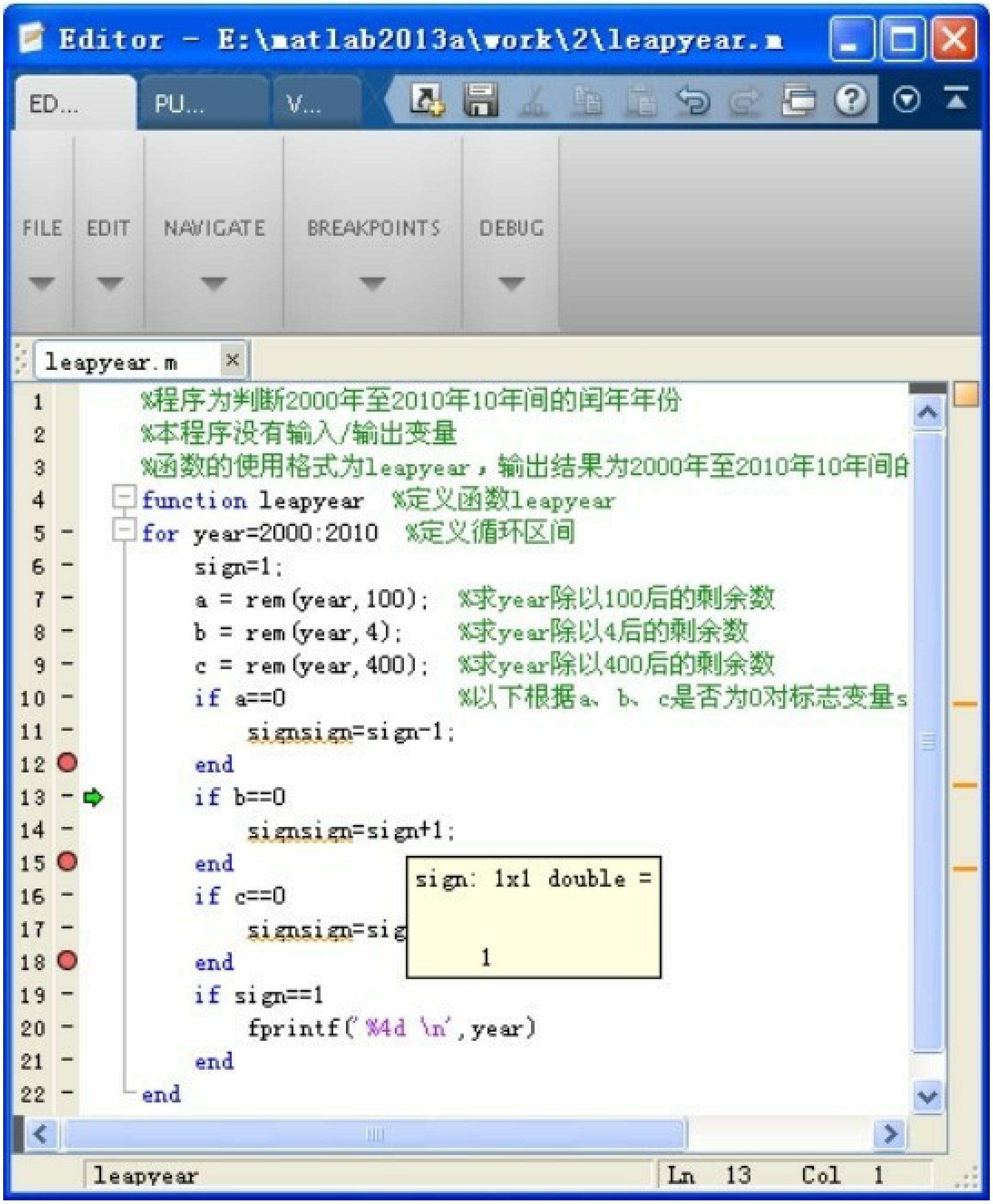
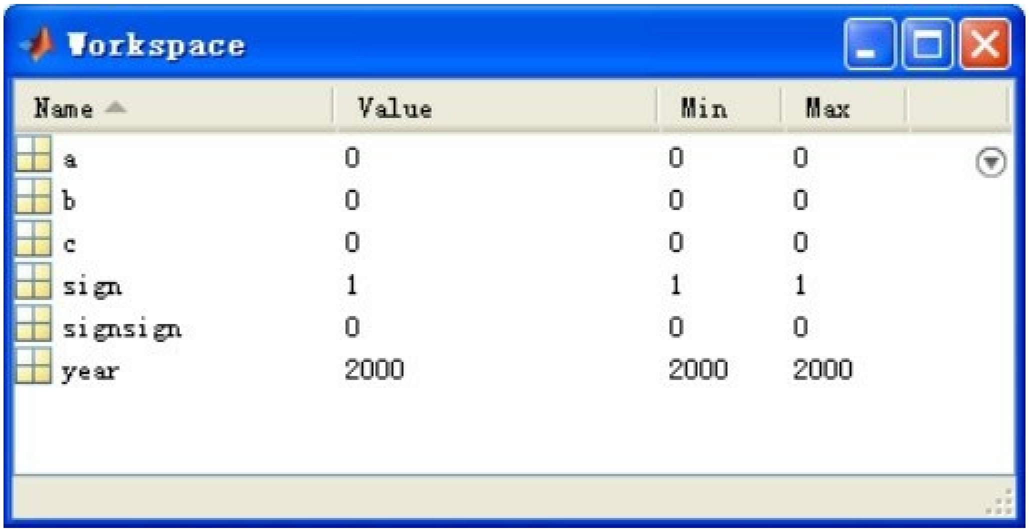


图3-4 用鼠标停留方法查看中间变量



Workspace

Name	Value	Min	Max
a	0	0	0
b	0	0	0
c	0	0	0
sign	1	1	1
signsign	0	0	0
year	2000	2000	2000

图3-5 查看workspace中所有中间变量的当前值

（9）修正代码。通过查看中间变量可知，在任何情况下sign的值都是1，此时调整修改代码程序如下所示。

```
>>%程序为判断2000年至2010年10年间的闰年年份
%本程序没有输入/输出变量
%函数的使用格式为leapyear，输出结果为2000年至2010年10年间的闰年年份
function leapyear
for year=2000:2010
  sign=0;
  a = rem(year,400);
  b = rem(year,4);
  c = rem(year,100);
  if a ==0
    sign=sign+1;
  end
  if b==0
    sign=sign+1;
```

```
    end
    if c==0
        sign=sign-1;
    end
    if sign==1
        fprintf('%4d \n',year)
    end
end
```

按F5键再次执行程序，得到的运行结果如下。

```
>> leapyear
2000
2004
2008
```

分析发现，结果正确，此时程序调试结束。

3.7 本章小结

MATLAB 语言称为第四代编程语言，程序简洁、可读性很强而且调试十分容易，是MATLAB的重要组成部分。

MATLAB为用户提供了非常方便易懂的程序设计方法，类似于其他的高级语言编程。本章侧重于MATLAB中最基础的程序设计，分别介绍了M文件、程序控制结构、数据的输入与输出、面向对象编程、程序优化及程序调试等内容。

第4章 人工神经网络概述

人工神经网络（Artificial Neural Networks，简写为ANN）也简称为神经网络（NN），它是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。

本章将简单介绍神经网络的基础知识。

学习目标：

·熟悉神经网络的概念

·掌握神经网络结构及特点

·掌握人工神经网络工具箱的函数和使用方法

4.1 人工神经网络

人工神经网络就是模拟人思维的第二种方式。这是一个非线性动力学系统，其特色在于信息的分布式存储和并行协同处理。虽然单个神经元的结构极其简单，功能有限，但大量神经元构成的网络系统所能实现的行为却是极其丰富多彩的。

4.1.1 人工神经网络的发展

1943年，心理学家W·Mcculloch和数理逻辑学家W·Pitts在分析、总结神经元基本特性的基础上首先提出神经元的数学模型。此模型沿用至今，并且直接影响着这一领域研究的进展。因此，他们两人可称为人工神经网络研究的先驱。

1945年，冯·诺依曼领导的设计小组试制成功存储程序式电子计算机，标志着电子计算机时代的开始。1948年，他在研究工作中比较了人脑结构与存储程序式计算机的根本区别，提出了以简单神经元构成的再生自动机网络结构。

但是，由于指令存储式计算机技术的发展非常迅速，迫使他放弃了神经网络研究的新途径，继续投身于指令存储式计算机技术的研究，并在此领域作出了巨大贡献。虽然，冯·诺依曼的名字是与普通计算机联系在一起的，但他也是人工神经网络研究的先驱之一。

20世纪50年代末，F·Rosenblatt设计制作了“感知机”，它是一种多层的神经网络。这项工作首次把人工神经网络的研究从理论探讨付诸工程实践。当时，世界上许多实验室仿效制作感知机，分别应用于文字识别、声音识别、声纳信号识别以及学习记忆问题的研究。

然而，这次人工神经网络的研究高潮未能持续很久，许多人陆续放弃了这方面的研究工作，这是因为当时数字计算机的发展处于全盛时期，许多人误以为数字计算机可以解决人工智能、模式识别、专家系统等方面的一切问题，使感知机的工作得不到重视。

当时的电子技术工艺水平比较落后，主要的元件是电子管或晶体管，利用它们制作的神经网络体积庞大，价格昂贵，要使其在规模上与真实的神经网络相似是完全不可能的。

在1968年一本名为《感知机》的著作中指出线性感知机功能是有限的，它不能解决如异感这样的基本问题，而且多层网络还不能找到有效的计算方法，这些论点促使大批研究人员对于人工神经网络的前景失去信心。20 世纪 60 年代末期，人工神经网络的研究进入了低潮。

在20世纪60年代初期，Widrow提出了自适应线性元件网络，这是一种连续取值的线性加权求和阈值网络。后来，在此基础上发展了非线性多层自适应网络。当时，这些工作虽未标出神经网络的名称，但实际上就是一种人工神经网络模型。

随着人们对感知机兴趣的衰退，神经网络的研究沉寂了相当长的时间。20 世纪 80 年代初期，模拟与数字混合的超大规模集成电路制作技术提高到新的水平，完全付诸实用化，此外，数字计算机的发展在若干应用领域遇到困难。

这一背景预示，向人工神经网络寻求出路的时机已经成熟。美国的物理学家 Hopfield于1982年和1984年在美国科学院院刊上发表了两篇关于人工神经网络研究的论文，引起了巨大的反响。人们重新认识到神经网络的威力以及付诸应用的现实性。随即，一大批学者和研究人员围绕着Hopfield提出的方法展开了进一步的工作，形成了人工神经网络的研究热潮。

4.1.2 人工神经网络研究内容

神经网络的研究内容相当广泛，反映了多学科交叉技术领域的特点。目前，主要的研究工作集中在以下几个方面。

1.生物原型研究

从生理学、心理学、解剖学、脑科学、病理学等生物科学方面研究神经细胞、神经网络、神经系统的生物原型结构及其功能机理。

2.建立理论模型

根据生物原型的研究，建立神经元、神经网络的理论模型，包括概念模型、知识模型、物理化学模型、数学模型等。

3.网络模型与算法研究

在理论模型研究的基础上构建具体的神经网络模型，以实现计算机模拟或准备制作硬件，包括网络学习算法的研究。这方面的工作也称为技术模型研究。

神经网络用到的算法就是向量乘法，并且广泛采用符号函数及其各种逼近。并行、容错、可以硬件实现以及自我学习特性，是神经网络的几个基本优点，也是神经网络计算方法与传统方法的区别所在。

4.人工神经网络应用系统

在网络模型与算法研究的基础上，利用人工神经网络组成实际的应用系统，例如，完成某种信号处理或模式识别的功能、构建专家系统、制成机器人等。

纵观当代新兴科学技术的发展历史，人类在征服宇宙空间、基本粒子，生命起源等科学技术领域的进程中历经了崎岖不平的道路。我们也会看到，探索人脑功能和神经网络的研究将伴随着对重重困难的克服而日新月异。

4.1.3 人工神经网络研究方向

神经网络的研究可以分为理论研究和应用研究两大方面。其中理论

研究又可分为以下两类。

（1）利用神经生理与认知科学研究人类思维以及智能机理。

（2）利用神经基础理论的研究成果，用数理方法探索功能更加完善、性能更加优越的神经网络模型，深入研究网络算法和性能，如稳定性、收敛性、容错性、鲁棒性等；开发新的网络数理理论，如神经网络动力学、非线性神经场等。

应用研究也可分为以下两类。

（1）神经网络的软件模拟和硬件实现的研究。

（2）神经网络在各个领域中应用的研究。这些领域主要包括：模式识别、信号处理、知识工程、专家系统、优化组合、机器人控制等。随着神经网络理论本身以及相关理论、相关技术的不断发展，神经网络的应用定将更加深入。

4.1.4 人工神经网络发展趋势

人工神经网络特有的非线性适应性信息处理能力，克服了传统人工智能方法对于直觉，如模式、语音识别、非结构化信息处理方面的缺陷，使之在神经专家系统、模式识别、智能控制、组合优化、预测等领域得到成功应用。

人工神经网络与其他传统方法相结合，推动人工智能和信息处理技术不断发展。近年来，人工神经网络正向模拟人类认知的道路上更加深入发展，与模糊系统、遗传算法、进化机制等结合，形成计算智能，成为人工智能的一个重要方向，将在实际应用中得到发展。

将信息几何应用于人工神经网络的研究，为人工神经网络的理论研究开辟了新的途径。神经计算机的研究发展很快，已有产品进入市场。光电结合的神经计算机为人工神经网络的发展提供了良好条件。

神经网络在很多领域已得到了很好的应用，但其需要研究的方面还

很多。其中，具有分布存储、并行处理、自学习、自组织以及非线性映射等优点的神经网络与其他技术的结合以及由此而来的混合方法和混合系统，已经成为一大研究热点。

由于其他方法也有它们各自的优点，所以将神经网络与其他方法相结合，取长补短，继而可以获得更好的应用效果。目前这方面工作有神经网络与模糊逻辑、专家系统、遗传算法、小波分析、混沌、粗集理论、分形理论、证据理论和灰色系统等的融合。

下面主要就神经网络与小波分析、混沌、粗集理论、分形理论的融合进行分析。

1.与小波分析的结合

1981年，法国地质学家Morlet在寻求地质数据时，通过对Fourier变换与加窗Fourier变换的异同、特点及函数构造进行创造性的研究，首次提出了“小波分析”的概念，建立了以他的名字命名的Morlet小波。1986年以来由于Y.Meyer、S.Mallat及I.Daubechies等人的奠基工作，小波分析迅速发展成为一门新兴学科。Meyer所著的“小波与算子”，Daubechies所著的“小波十讲”是小波研究领域最权威的著作。

小波变换是对Fourier分析方法的突破。它不但在时域和频域同时具有良好的局部化性质，而且对低频信号在频域和对高频信号在时域里都有很好的分辨率，从而可以聚集到对象的任意细节。小波分析相当于一个数学显微镜，具有放大、缩小和平移功能，通过检查不同放大倍数下的变化来研究信号的动态特性。因此，小波分析已成为地球物理、信号处理、图像处理、理论物理等诸多领域的强有力工具。

小波神经网络将小波变换良好的时频局域化特性和神经网络的自学习功能相结合，因而具有较强的逼近能力和容错能力。在结合方法上，可以将小波函数作为传递函数构造神经网络形成小波网络，或者小波变换作为前馈神经网络的输入前置处理工具，即以小波变换的多分辨率特性对过程状态信号进行处理，实现信噪分离，并提取出对加工误差影响

最大的状态特性，作为神经网络的输入。

小波神经网络在电机故障诊断、高压电网故障信号处理与保护研究、轴承等机械故障诊断以及许多方面都有应用，将小波神经网络用于感应伺服电机的智能控制，使该系统具有良好的跟踪控制性能，以及好的鲁棒性，利用小波包神经网络进行心血管疾病的智能诊断，小波层进行时频域的自适应特征提取，前向神经网络用来进行分类，正确分类率达到94%。

小波神经网络虽然应用于很多方面，但仍存在一些不足。从提取精度和小波变换实时性的要求出发，有必要根据实际情况构造一些适应应用需求的特殊小波基，以便在应用中取得更好的效果。另外，在应用中的实时性要求，也需要结合DSP的发展，开发专门的处理芯片，从而满足这方面的要求。

2.混沌神经网络

混沌第一个定义是20世纪70年代才被T.Y.Li和J.A.Yorke第一次提出的。由于它具有广泛的应用价值，自它出现以来就受到各方面的普遍关注。混沌是一种确定的系统中出现的无规则的运动，混沌是存在于非线性系统中的一种较为普遍的现象，混沌运动具有遍历性、随机性等特点，能在一定的范围内按其自身规律不重复地遍历所有状态。混沌理论所决定的是非线性动力学混沌，目的是揭示貌似随机的现象背后可能隐藏的简单规律，以求发现一大类复杂问题普遍遵循的共同规律。

1990年K.Aihara、T.Takabe和M.Toyoda等人根据生物神经元的混沌特性首次提出混沌神经网络模型，将混沌学引入神经网络中，使得人工神经网络具有混沌行为，更加接近实际的人脑神经网络，因而混沌神经网络被认为是可实现其真实世界计算的智能信息处理系统之一，成为神经网络的主要研究方向之一。

与常规的离散型Hopfield神经网络相比较，混沌神经网络具有更丰富的非线性动力学特性，主要表现如下：在神经网络中引入混沌动力学

行为；混沌神经网络的同步特性；混沌神经网络的吸引子。

当神经网络实际应用中，网络输入发生较大变异时，应用网络的固有容错能力往往感到不足，经常会发生失忆现象。混沌神经网络动态记忆属于确定性动力学运动，记忆发生在混沌吸引子的轨迹上，通过不断地运动（回忆过程），一一联想到记忆模式，特别对于那些状态空间分布得较接近或者发生部分重叠的记忆模式，混沌神经网络总能通过动态联想记忆加以重现和辨识，而不发生混淆，这是混沌神经网络所特有的性能，它将大大改善Hopfield 神经网络的记忆能力。混沌吸引子的吸引域存在，形成了混沌神经网络固有容错功能。这将对复杂的模式识别、图像处理等工程应用发挥重要作用。

混沌神经网络受到关注的另一个原因是混沌存在于生物体真实神经元及神经网络中，并且起到一定的作用，动物学的电生理实验已证实了这一点。

混沌神经网络由于其复杂的动力学特性，在动态联想记忆、系统优化、信息处理、人工智能等领域受到人们极大的关注。针对混沌神经网络具有联想记忆功能，但其搜索过程不稳定，提出了一种控制方法可以对混沌神经网络中的混沌现象进行控制。研究了混沌神经网络在组合优化问题中的应用。

为了更好地应用混沌神经网络的动力学特性，并对其存在的混沌现象进行有效的控制，仍需要对混沌神经网络的结构进行进一步的改进和调整，以及对混沌神经网络算法的进一步研究。

3.基于粗集理论

粗糙集（Rough sets）理论是1982 年由波兰华沙理工大学教授Z.Pawlak 首先提出，它是一个分析数据的数学理论，研究不完整数据、不精确知识的表达、学习、归纳等方法。粗糙集理论是一种新的处理模糊和不确定性知识的数学工具，其主要思想就是在保持分类能力不变的前提下，通过知识约简，导出问题的决策或分类规则。目前，粗糙集理

论已被成功应用于机器学习、决策分析、过程控制、模式识别与数据挖掘等领域。

目前粗集与神经网络的结合已应用于语音识别、专家系统、数据挖掘、故障诊断等领域，将神经网络和粗集用于声源位置的自动识别，将神经网络和粗集用于专家系统的知识获取中，取得比传统专家系统更好的效果，其中粗集进行不确定和不精确数据的处理，神经网络进行分类工作。

虽然粗集与神经网络的结合已应用于许多领域的研究，为使这一方法发挥更大的作用还需考虑如下问题：模拟人类抽象逻辑思维的粗集理论方法和模拟形象直觉思维的神经网络方法更加有效的结合；二者集成的软件和硬件平台的开发，提高其实用性。

4.与分形理论的结合

自从美国哈佛大学数学系教授Benoit B.Mandelbrot于20世纪70年代中期引入分形这一概念，分形几何学（Fractal geometry）已经发展成为科学的方法论——分形理论，且被誉为开创了 20 世纪数学重要阶段。现已被广泛应用于自然科学和社会科学的几乎所有领域，成为现今国际上许多学科的前沿研究课题之一。

由于在许多学科中的迅速发展，分形已成为一门描述自然界中许多不规则事物的规律性的学科。它已被广泛应用在生物学、地球地理学、天文学、计算机图形学等各个领域。

用分形理论来解释自然界中那些不规则、不稳定和具有高度复杂结构的现象，可以收到显著的效果，而将神经网络与分形理论相结合，充分利用神经网络非线性映射、计算能力、自适应等优点，可以取得更好的效果。

分形神经网络的应用领域有图像识别、图像编码、图像压缩，以及机械设备系统的故障诊断等。分形图像压缩/解压缩方法有着高压缩率和低遗失率的优点，但运算能力不强，由于神经网络具有并行运算的特

点，将神经网络用于分形图像压缩/解压缩中，提高了原有方法的运算能力。

将神经网络与分形相结合用于果实形状的识别，首先利用分形得到几种水果轮廓数据的不规则性，然后利用3层神经网络对这些数据进行辨识，继而对其不规则性进行评价。

分形神经网络已得到了许多应用，但仍有些问题值得进一步研究：分形维数的物理意义；分形的计算机仿真和实际应用研究。随着研究的不断深入，分形神经网络必将得到不断的完善，并取得更好的应用效果。

4.2 神经元

人工神经网络是一种运算模型，由大量的神经元及其相互联接构成。每个节点代表一种特定的输出函数，称为激励函数（activation function）。每两个神经元之间的连接都代表一个对于通过该连接信号的加权值，称之为权重，这相当于人工神经网络的记忆。

网络的输出则依网络的连接方式、权重值和激励函数的不同而不同。而网络自身通常都是对自然界某种算法或者函数的逼近，也可能是对一种逻辑策略的表达。

4.2.1 神经元细胞

神经元是神经系统的结构和功能的基本单位。它有突起，突起可延伸至全身各器官和组织中，突起分为树突和轴突。一个神经元有轴突，可以把兴奋从胞体传送到另一个神经元或其他组织，但是有许多树突接受刺激并将兴奋传入细胞体。

一个神经元细胞由细胞体、突触、轴突和树突组成，结构如图4-1所示。

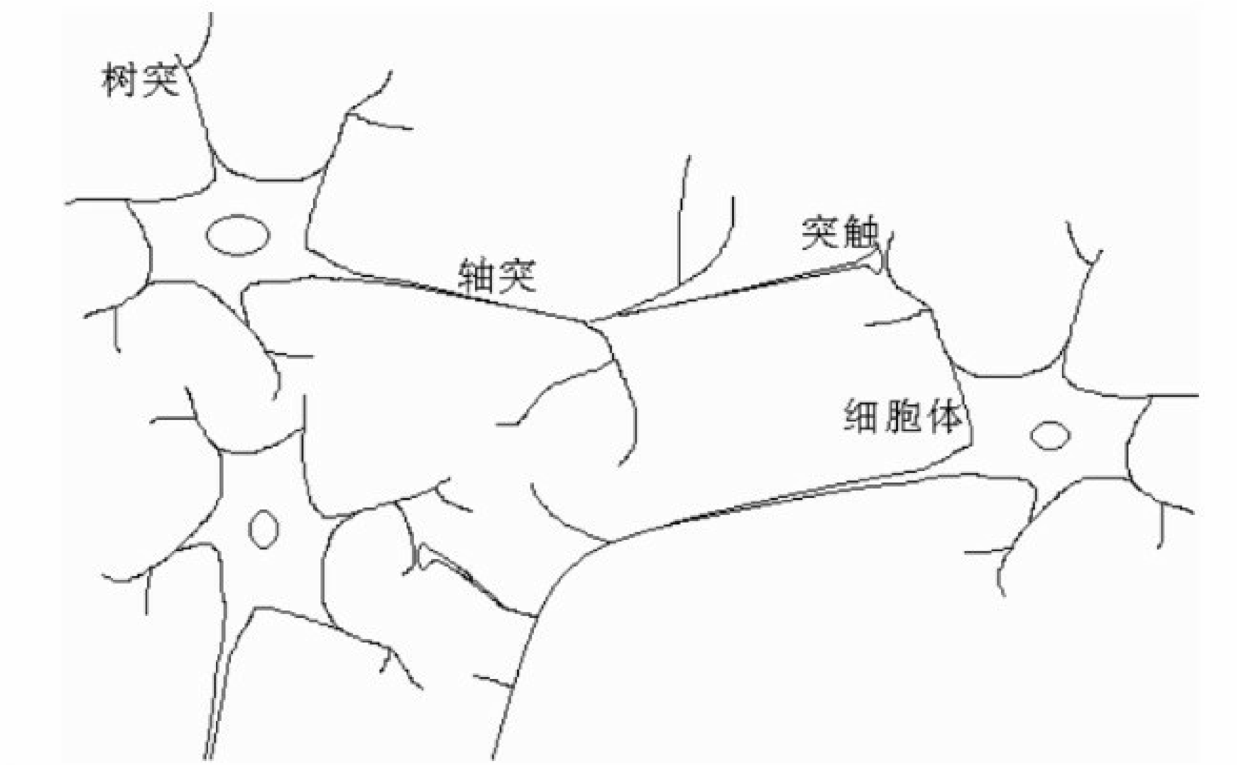


图4-1 神经元结构

细胞体：在脑和脊髓的灰质及神经节内，其形态各异，常见的形态为星形、锥体形、梨形和圆球形状等。

树突：从胞体发出的一至多个突起，呈放射状。

轴突：细胞体突起的最长的外伸管状纤维。

突触：神经元与神经元之间的连接点。它是神经元之间传递信息的关键性结构。突触可分两类，即化学性突触和电突触。

神经元之间通过突触传递信息。当神经冲动传至突触前膜时，突触小泡移向突触前膜，以胞吐方式释放小泡内的神经递质，其中部分神经递质与突触后膜上的相应受体结合，后膜内外两侧的离子分布状况发生改变，呈现兴奋性或抑制性变化，从而影响突触后神经元（或效应细胞）的活动。

使突触后膜发生兴奋的突触，称兴奋性突触，而使后膜发生抑制的称抑制性突触。

突触的兴奋或抑制决定于神经递质及其受体的种类。由于一个神经

元通常有许多突触，其中有些是兴奋性的，有些是抑制性的。如果兴奋性突触活动强度总和超过抑制性突触活动强度总和，并达到一定阈值，就能使该神经元的轴突起始段发生动作电位，产生神经冲动。出现神经冲动时，该神经元呈现兴奋，反之，则表现为抑制。

4.2.2 MP模型

1943 年，美国心理学家 MeCulloch 和数学家 Pitts 共同提出“模拟生物神经元”，被称为 MP 的人工神经元，人工神经网络是由大量的处理单元即人工神经元广泛互连组成的网络。

网络的信息处理功能由神经元间相互作用来实现，知识与信息的存储表现为网络元件间分布式的物理联系，网络的学习和识别取决于各种神经元件连接权系数的动态演化过程。

其主要特征为连续时间非线性动力学、网络的全局作用、大规模的并行分布处理和联想学习能力。MP 的人工神经元一般是一个多输入多输出的非线性部件，其基本结构如图4-2所示。

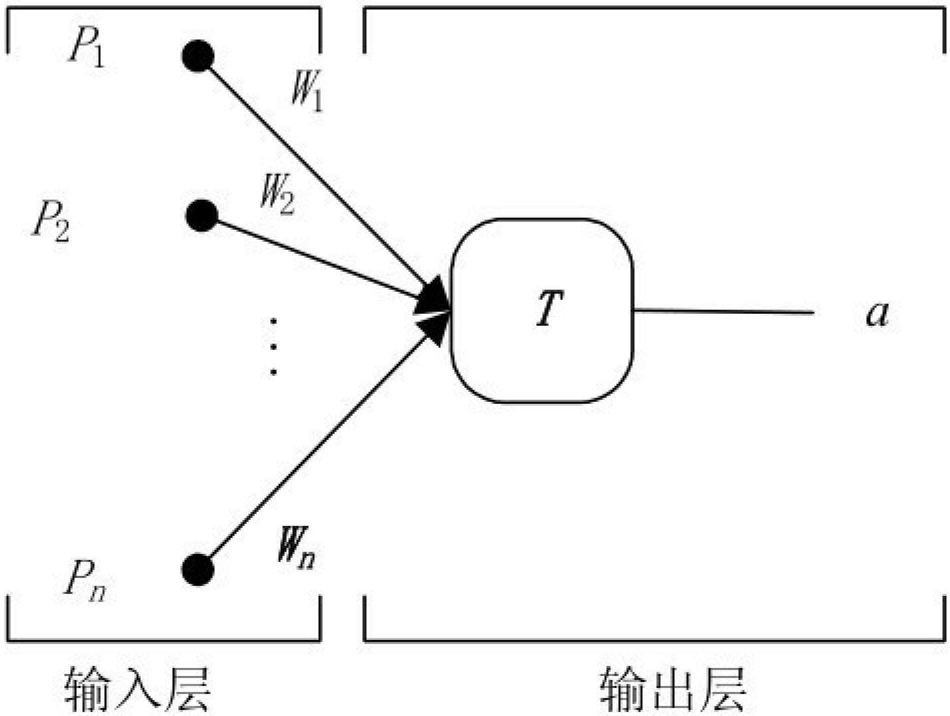


图4-2 MP模型

图4-2中P为神经元的输入，W为神经元与输入相对应的权值，T为神经元兴奋阈值，a为神经元的输出。神经元的输出值如下所示。

$$a=\begin{cases}1, & 若\sum_{i=1}^{n}W_iP_i \geqslant T\\ 0, & 若\sum_{i=1}^{n}W_iP_i < T\end{cases}$$

上式中，n为神经元输入个数，P_i表示神经元的第i个输入，W_i表示神经元的第i个输入权值，a表示神经元的输出，T表示神经元的阈值。

因为通过简单的计算说明了人工神经网络可产生相当复杂的行为，该模型从而引起极大的轰动。但它是一种静态神经元，即结构固定，权值无法调节，因此缺乏一个关键性的要素，即学习能力。

单个MP神经元模型可以实现与、或、与非、或非等二值逻辑运算（但不能实现异或运算）。

4.2.3 一般神经元模型

由于MP模型过于简单，而且权值不能学习，因此需要更复杂的、灵活性更高的神经元模型。一个具有n个输入的神经元模型如图4-3所示，其中每个输入与对应权值W的乘积之和与阈值一起作为传递函数 f 的输入。在神经元中，可以使用任何可微的函数作为它的传递函数。

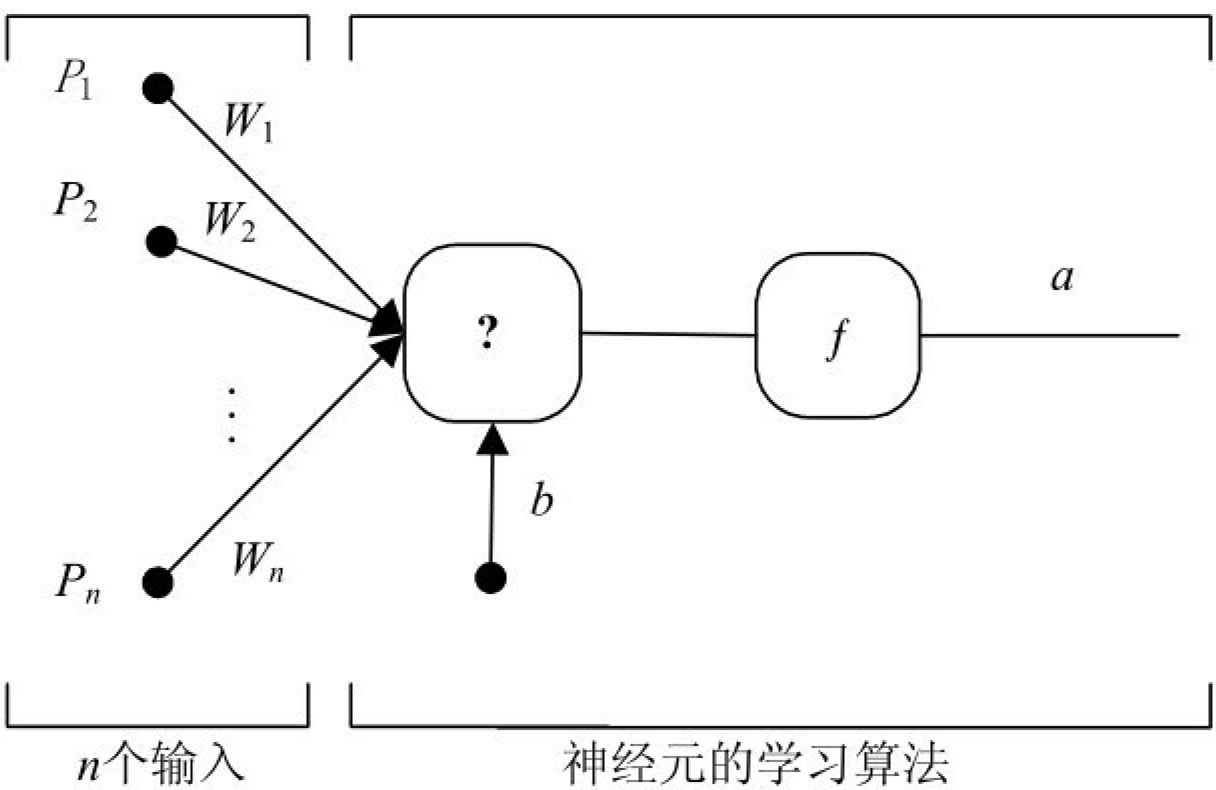


图4-3 神经元模型

与MP模型类似，P表示神经元的输入，W表示与神经元输入值对应的权值，b 是神经元的偏置，n表示输入向量的个数，f表示神经元的传递函数，a为神经元的输出，计算公式如下。

$$a = f(\sum_{i=1}^{n} W_i P_i + b)$$

传递函数可以是线性的，也可以是非线性的。常用的传递函数有以下一些类型。

1.符号函数

函数表达式为

$$f(x) = Sgn(x) = \begin{cases} 1, x \geqslant 0 \\ -1, x < 0 \end{cases}$$

其对应的图形如图4-4所示。

2.线性函数

如果传递函数采用线性函数，则神经元输出a为。

f (x)=x

线性函数的曲线如图 4-5 所示。该激活函数常用于实现函数逼近的神经网络的输出层神经元。

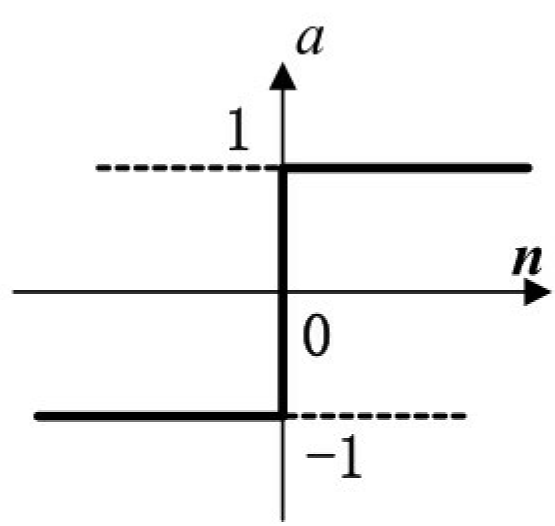


图4-4 符号函数

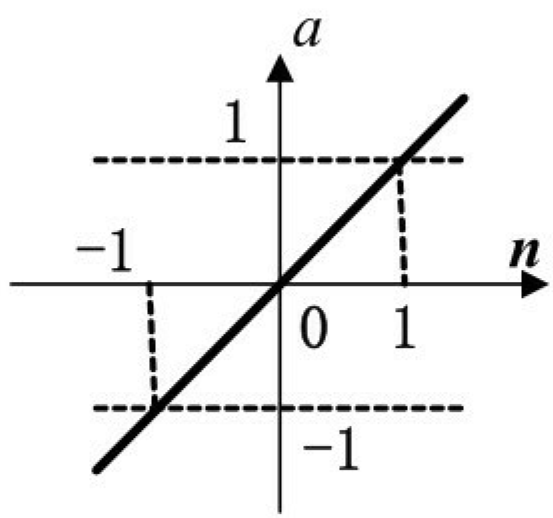


图4-5 线性函数曲线

3.Sigmoidal函数

Sigmoidal函数也叫S函数，是一类非常重要的激活函数，无论神经网络用于分类、函数逼近或优化，Sigmoidal函数都是常用的激活函数。Sigmoidal函数的表达式为

$$f(x)=\frac{1}{1+\exp(-ax)}$$

对数S形函数图形如图4-6所示。

4.高斯函数

高斯函数（也称钟形函数）也是极为重要的一类激活函数，常用于径向基函数神经网络（RBF 网络）。高斯函数的表达式为

$$a=f(x)=e^{-\frac{u^2}{\delta^2}}$$

其中参数δ被称为高斯函数的宽度或扩展常数。δ越大，函数形状就越平坦；反之，δ越小，函数形状就越陡峭。

高斯函数的曲线如图4-7所示。

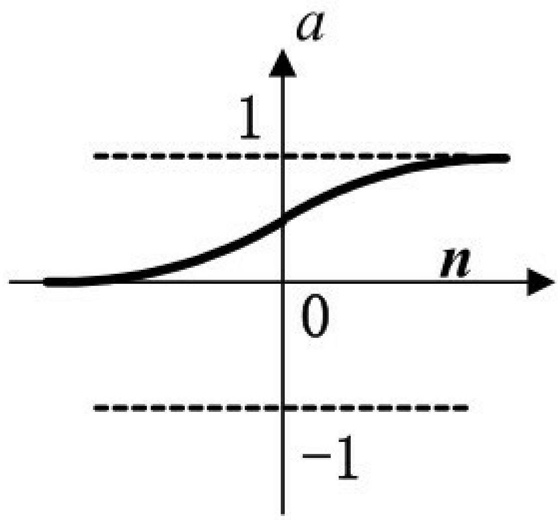


图4-6 对数S 形函数曲线

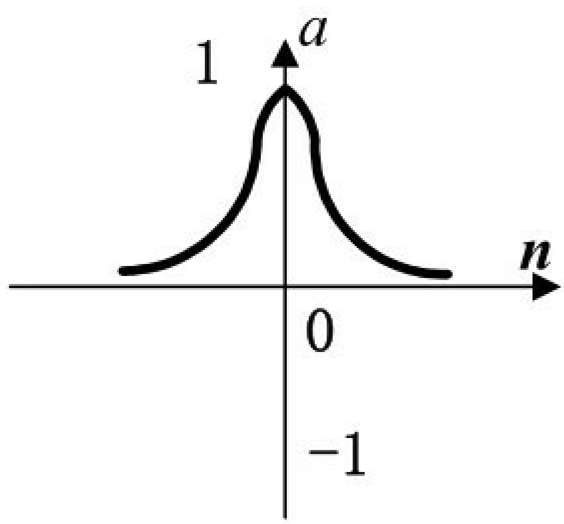


图4-7 高斯函数曲线

4.3 神经网络的结构及学习

人工神经网络的构筑理念是受到生物（人或其他动物）神经网络功能的运作启发而产生的。人工神经网络通常是通过一个基于数学统计学类型的学习方法（Learning Method）得以优化，所以人工神经网络也是数学统计学方法的一种实际应用。

通过统计学的标准数学方法，我们能够得到大量的可以用函数来表达的局部结构空间，另一方面在人工智能学的人工感知领域，我们通过数学统计学的应用可以来做人工感知方面的决定问题（也就是说通过统计学的方法，人工神经网络能够类似人一样具有简单的决定能力和简单的判断能力），这种方法比起正式的逻辑学推理演算更具有优势。

4.3.1 神经网络结构

人工神经网络模型主要考虑网络连接的拓扑结构、神经元的特征、学习规则等。目前，已有近40种神经网络模型，其中有反传网络、感知器、自组织映射、Hopfield网络、波耳兹曼机、适应谐振理论等。根据连接的拓扑结构，神经网络模型可以分为以下两种。

1.前向网络

网络中各个神经元接受前一级的输入，并输出到下一级，网络中没有反馈，可以用一个有向无环路图表示。这种网络实现信号从输入空间到输出空间的变换，它的信息处理能力来自于简单非线性函数的多次复合。

图4-8所示为两层前向神经网络，该网络只有输入层和输出层，其中x为输入，W为权值，y为输出。输出层神经元为计算节点，其传递函

数取符号函数f。该网络一般用于线性分类。

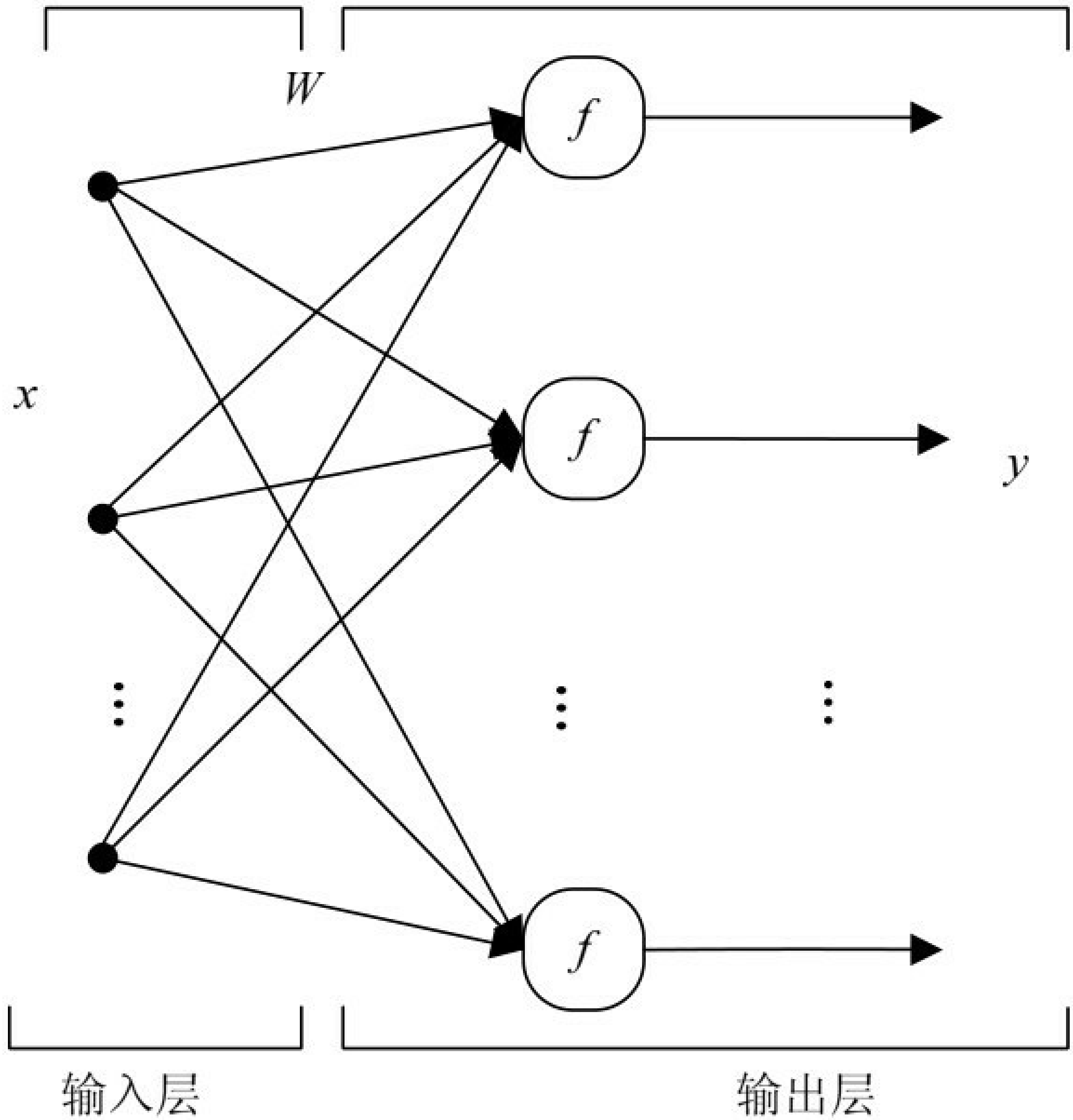


图4-8 两层前向神经网络

图 4-9 所示为多层前向神经网络，该网络有一个输入层、一个输出层和多个隐含层，其中隐含层和输出层神经元为计算节点。多层前向神经网络传递函数可以取多种形式。如果所有的计算节点都取符号函数，则网络称为多层离散感知器。

前向网络结构简单，易于实现。反传网络是一种典型的前向网络。

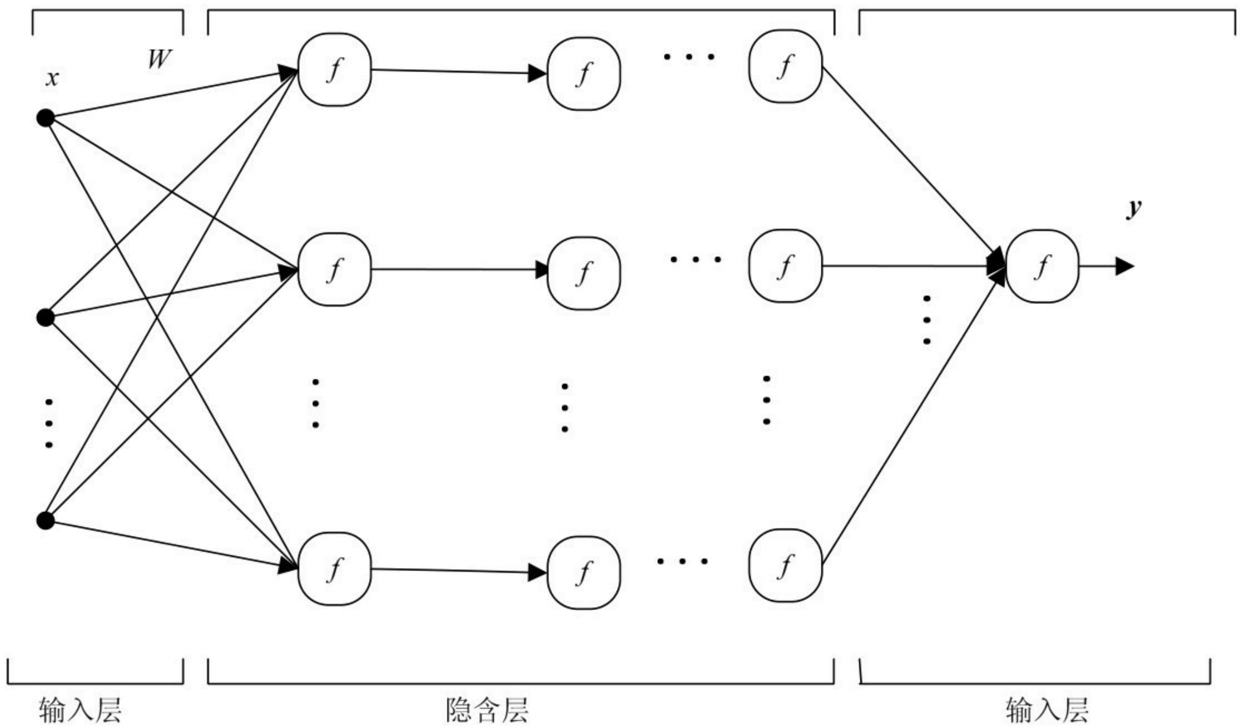


图4-9 多层前向神经网络

2.反馈网络

网络内神经元间有反馈，可以用一个无向的完备图表示。这种神经网络的信息处理是状态的变换，可以用动力学系统理论处理。系统的稳定性与联想记忆功能有密切关系。Hopfield网络、波耳兹曼机均属于这种类型。

以两层前馈神经网络模型（输入层为n个神经元）为例，反馈神经网络结构如图4-10所示。

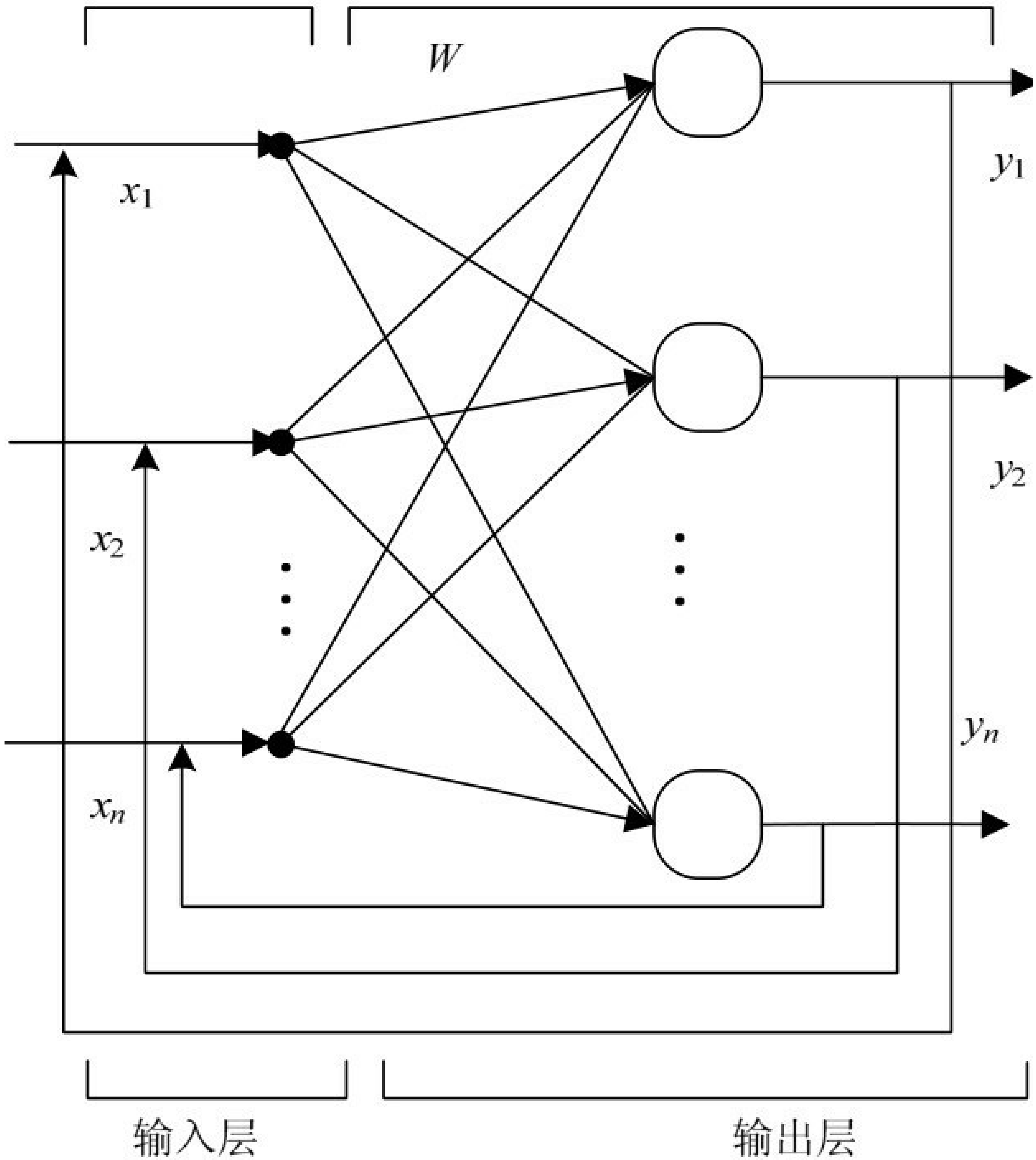


图4-10 反馈神经网络的结构

4.3.2 神经网络学习

学习是神经网络研究的一个重要内容，它的适应性是通过学习实现的。根据环境的变化，对权值进行调整，改善系统的行为。

由Hebb提出的Hebbian学习规则为神经网络的学习算法奠定了基础。Hebbian规则认为学习过程最终发生在神经元之间的突触部位，突触的联系强度随着突触前后神经元的活动而变化。

在此基础上，人们提出了各种学习规则和算法，以适应不同网络模型的需要。有效的学习算法，使得神经网络能够通过连接权值的调整，构造客观世界的内在表示，形成具有特色的信息处理方法，信息存储和处理体现在网络的连接中。

根据学习环境不同，神经网络的学习方式可分为监督学习和非监督学习。在监督学习中，将训练样本的数据加到网络输入端，同时将相应的期望输出与网络输出相比较，得到误差信号，以此控制权值连接强度的调整，经多次训练后收敛到一个确定的权值。

当样本情况发生变化时，经学习可以修改权值以适应新的环境。使用监督学习的神经网络模型有反传网络、感知器等。非监督学习时，事先不给定标准样本，直接将网络置于环境之中，学习阶段与工作阶段成为一体。

此时，学习规律的变化服从连接权值的演变方程。非监督学习最简单的例子是Hebbian学习规则。竞争学习规则是一个更复杂的非监督学习的例子，它是根据已建立的聚类进行权值调整。自组织映射、适应谐振理论网络等都是与竞争学习有关的典型模型。

1.Hebbian学习规则

1949年，心理学家D.O.Hebb最早提出了关于神经网络学习机理的“突触修正”的假设。该假设指出，当神经元的突触前膜电位与后膜电位同时为正时，突触传导增强，当前膜电位与后膜电位正负相反时，突触传导减弱。

也就是说，当神经元i与神经元j同时处于兴奋状态时，两者之间的

连接强度应增强。根据该假设定义的权值调整方法，称为Hebbian学习规则。

在 Hebbian 学习规则中，学习信号简单地等于神经元的输出。假定神经元的当前的输入为 $x=(x_1,x_2,\ldots,x_n)^T$，权值为w(t)，输出为 $y=f(w(t)^T\bullet x)$，则权值调节量为

$$\Delta w(t)=\eta yx$$

上式表明，权值调整量与输入输出的乘积成正比。显然，经常出现的输入模式将对权向量有最大的影响。在这种情况下，Hebbian学习规则需预先设置权饱和值，以防止输入和输出正负始终一致时出现权值无约束增长。

神经元权值修复公式为

$$w(t+1)=w(t)+\eta yx$$

神经元的初始权值一般取零附近的随机值，激活函数f可以取任意形式。式中Δw(t)可理解为样本x对当前权值的影响。

2.感知器学习规则

1958 年，美国学者 Frank Rosenblatt 首次定义了一个具有单层计算单元的神经网络结构，称为感知器（Perceptron）。感知器的学习规则规定，学习信号等于神经元期望输出（教师信号）与实际输出之差。

对样本输入x，假定神经元的期望输出为d，当前输出为y，而神经元的传递函数取符号函数。则感知器学习规则中，权值调整量为

$$\Delta w(t)=e(t)x$$

其中误差信号为

$$e(t) = d - y = d - Sgn(w^T x)$$

3.δ(Delta)学习规则

1986年，认知心理学家McClelland和Rumelhart在神经网络训练中引入了δ规则，δ学习规则也称梯度法或最速下降法，是最常用的神经网络学习算法。

4.Widrow-Hoff学习规则

1962 年，Bernard Widrow和Marcian Hoff 提出了Widrow-Hoff学习规则，又称为最小均方规则（LMS）。Widrow-Hoff学习规则与δ学习规则的推导类似，但该学习规则也可用于神经元传递函数取符号函数的情形。

4.4 MATLAB神经网络工具箱

MATLAB和Simulink包含进行神经网络应用设计和分析的许多工具箱函数。目前最新的神经网络工具箱几乎完整地概括了现有的神经网络的新成果。对于各种网络模型，神经网络工具箱集成了多种学习算法，为用户提供了极大的方便。

4.4.1 神经网络工具箱函数

神经网络理论的初学者可以利用该工具箱来深刻理解各种算法的内在实质；神经网络的应用者对数神经元层即使不了解算法的本质，也可以直接应用功能，通过丰富的函数来实现自己的想法；对研究者而言，该工具箱强大的扩充功能将令其工作起来游刃有余。最关键的是丰富的函数可以节约大量的编程时间。

神经网络工具箱几乎包括了现有神经网络的最新成果，神经网络工具箱模型包括如下内容。

·感知器

·线性网络

·BP网络

·径向基函数网络

·竞争型神经网络

·自组织网络和学习向量量化网络

·反馈网络

本节主要介绍通用的神经网络工具函数，对它们的使用方法、注意事项等进行说明。表4-1列出了神经网络中一些比较重要的通用函数。

表4-1 神经网络工具函数

	名称	用途	名称	用途
创建函数	newp	创建感知器网络	newlind	设计一线性层
	newlin	创建一线性层	newff	创建一前馈 BP 网络
	newcf	创建一多层前馈 BP 网络	newfftd	创建一前馈输入延迟 BP 网络
	newrb	设计一径向基网络	newrbe	设计一严格的径向基网络
	newgrnn	设计一广义回归神经网络	newpnn	设计一概率神经网络
	newc	创建一竞争层	newsom	创建一自组织特征映射
	newhop	创建一 Hopfield 递归网络	newelm	创建一 Elman 递归网络
应用函数	sim	仿真一个神经网络	init	初始化一个神经网络
	adapt	神经网络的自适应化	train	训练一个神经网络
学习函数	learnp	感知器学习函数	learnpn	标准感知器学习函数
	learnwh	Widrow_Hoff 学习规则	learngd	BP 学习规则
	learngdm	带动量项的 BP 学习规则	learnk	Kohonen 权学习函数
	learncon	Conscience 阈值学习函数	learnsom	自组织映射权学习函数
训练函数	trainwb	网络权与阈值的训练函数	traingdm	梯度下降 w/动量的 BP 算法训练函数
	traingd	梯度下降的 BP 算法训练函数	traingda	梯度下降 w/自适应 lr 的 BP 算法训练函数
	traingdx	梯度下降 w/动量和自适应 lr 的 BP 算法训练函数	trainlm	Levenberg_Marquardt 的 BP 算法训练函数
绘图函数	plotes	绘制误差曲面	plotsom	绘制自组织映射图
	plotep	绘制权和阈值在误差曲面上的位置		

4.4.2 神经网络工具箱的使用

首先在MATLAB命令窗口输入nnstart，可以打开MATLAB提供的神经网络图形用户界面，如图4-11所示。

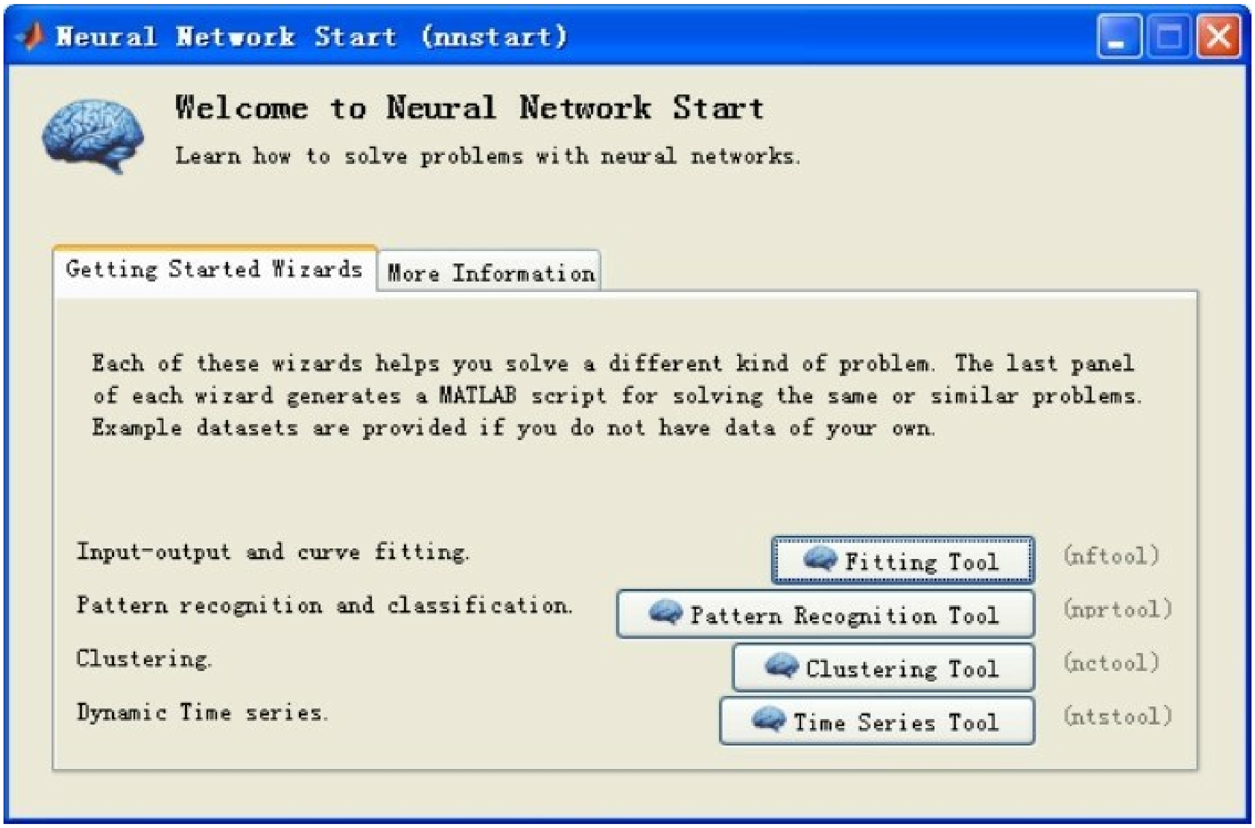


图4-11 神经网络图形用户界面

单击神经网络图形用户界面的“Fitting Tool”按钮，打开神经网络工具箱，如图 4-12所示。读者也可以在MATLAB命令窗口直接输入nftool直接打开神经网络工具箱。

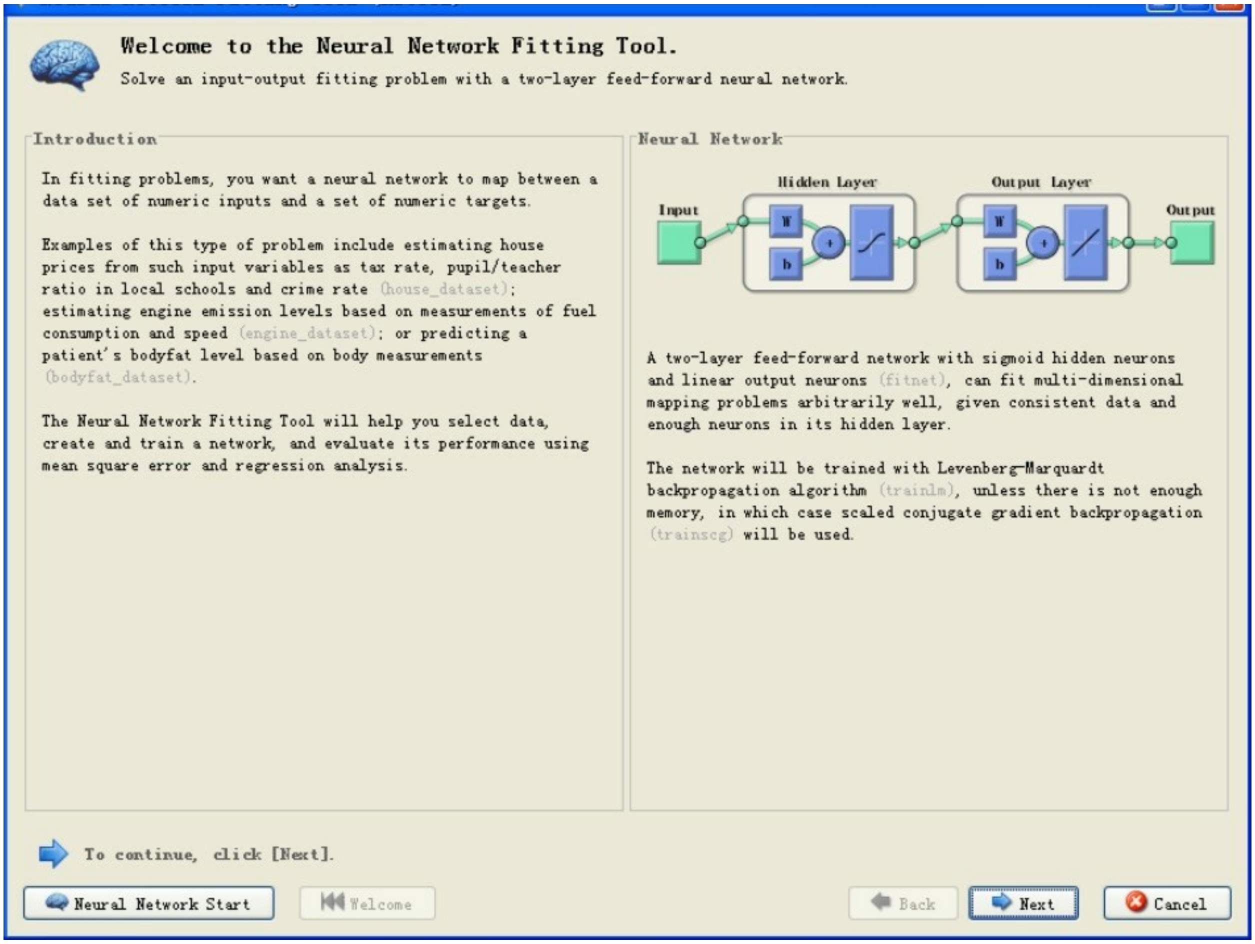


图4-12 神经网络工具箱

单击“Next”按钮，出现图4-13所示界面。读者可以在“Input”和“Targets”选项框分别选择神经网络的输入和目标值，也可以选择“Load Example Data Set”导入MATLAB提供的神经网络示例。

选择“Load Example Data Set”后出现图4-14 所示界面。

选择“House Pricing”后单击“Import”按钮，回到输入值及目标值选择界面，继续单击“Next”按钮，得到图4-15所示图形。在图中可以选择训练和测试的样本比例。

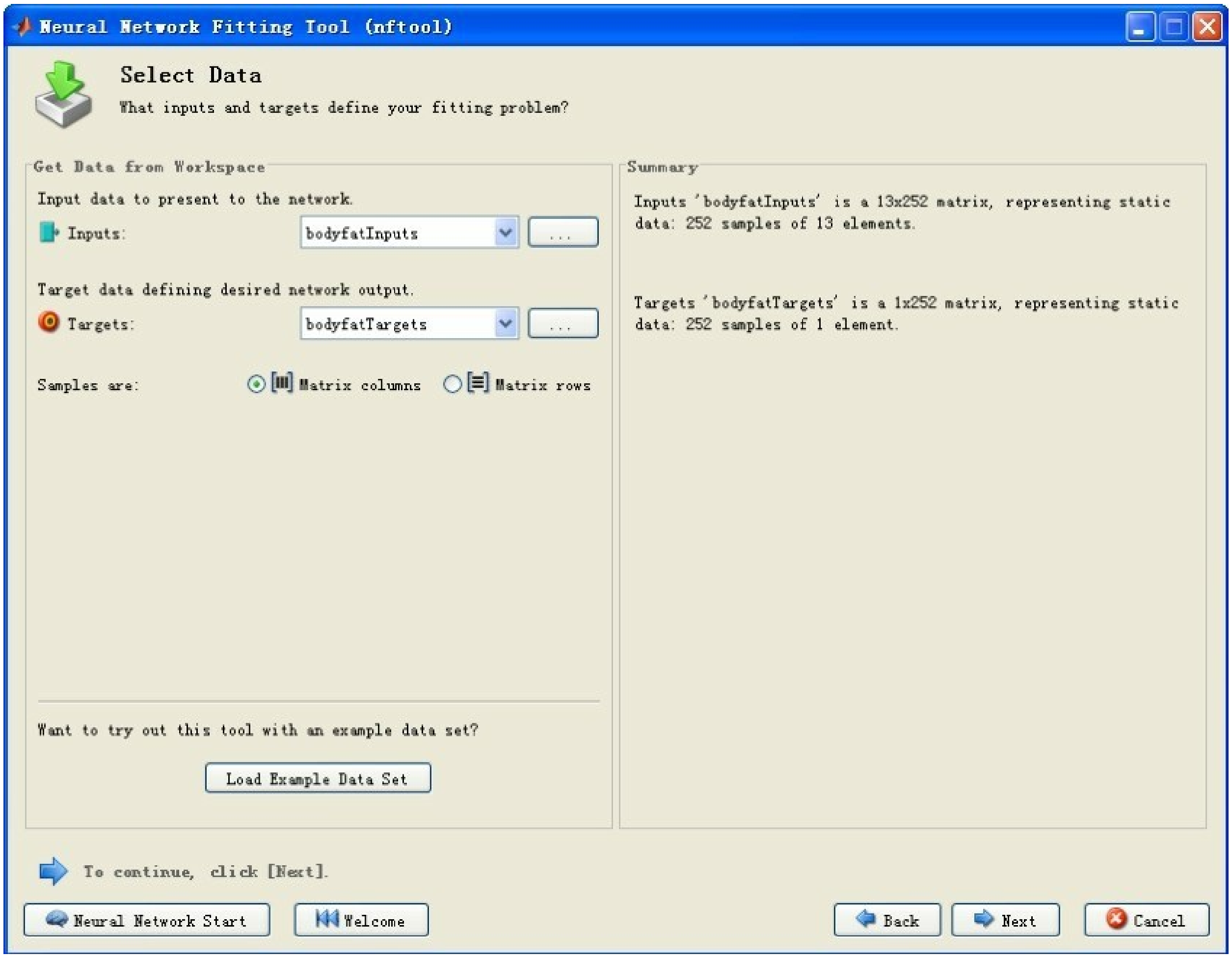


图4-13 输入值及目标值选择界面

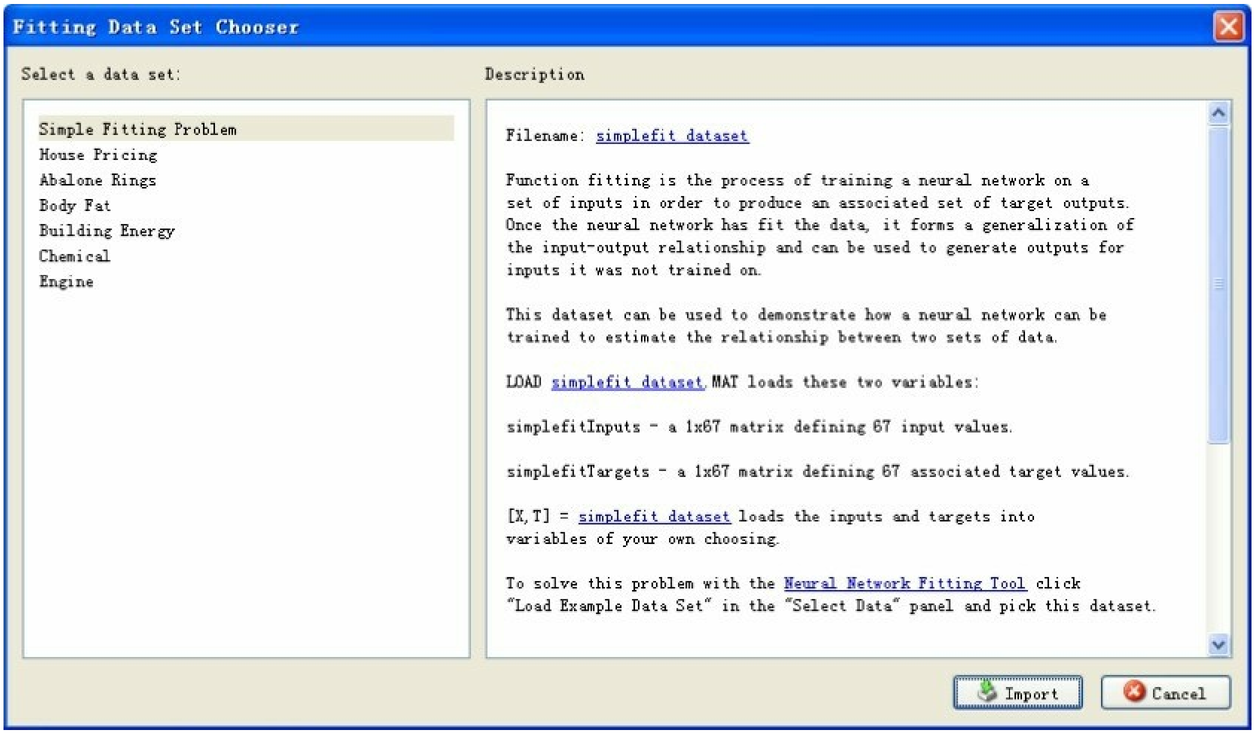


图4-14 MATLAB示例输入选择界面

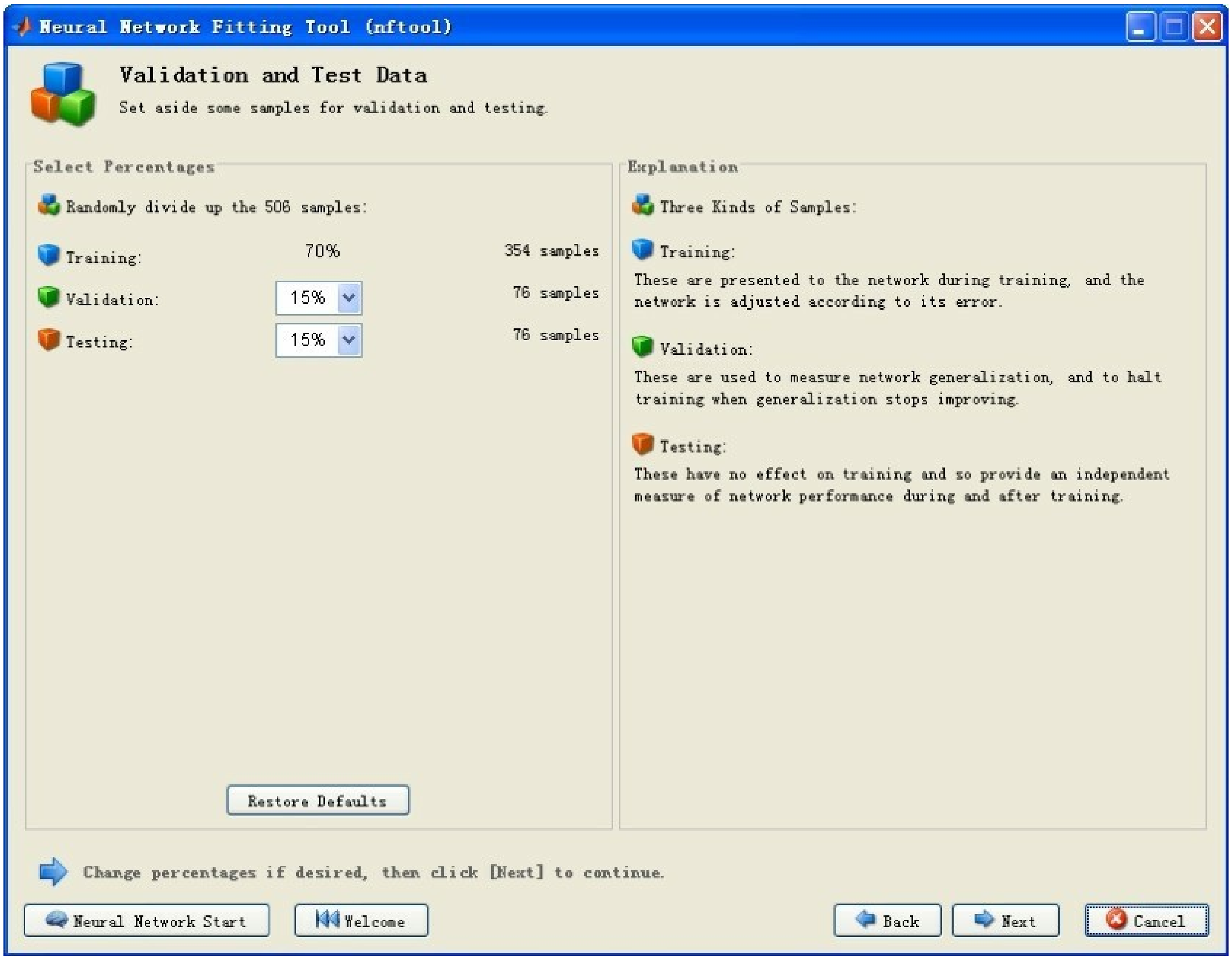


图4-15 样本比例选择界面

选择完成后单击“Next”按钮，建立图4-16所示神经网络结构图。本神经网络共3层，分别为输入层、输出层和隐含层。其中，隐含层神经元数目可以在图中修改。

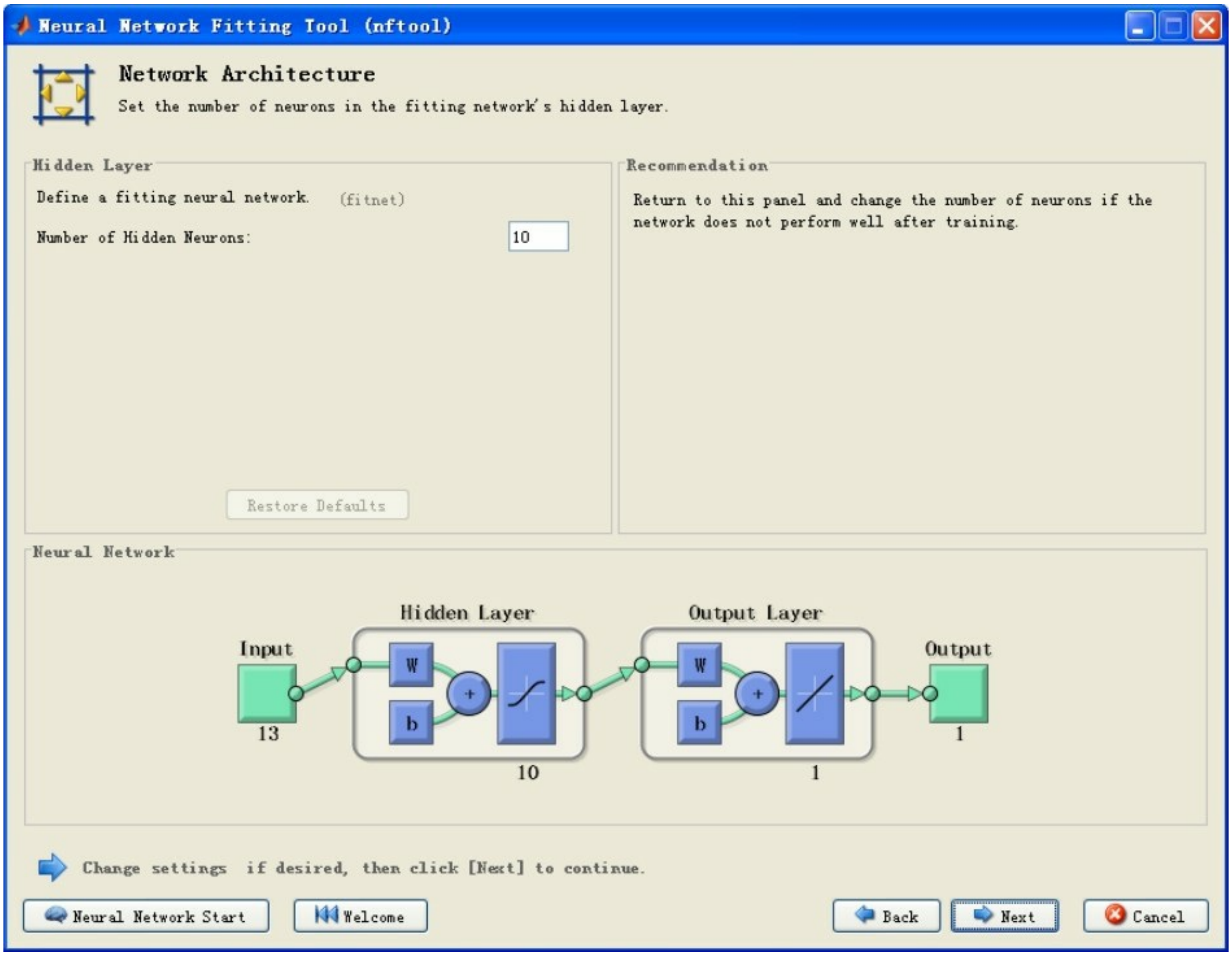


图4-16 神经网络隐含层神经元数目选择

确定神经网络隐含层神经元后，单击“Next”按钮，确定神经网络结构，在出现的界面中可以选择是否训练神经网络，如图4-17所示。

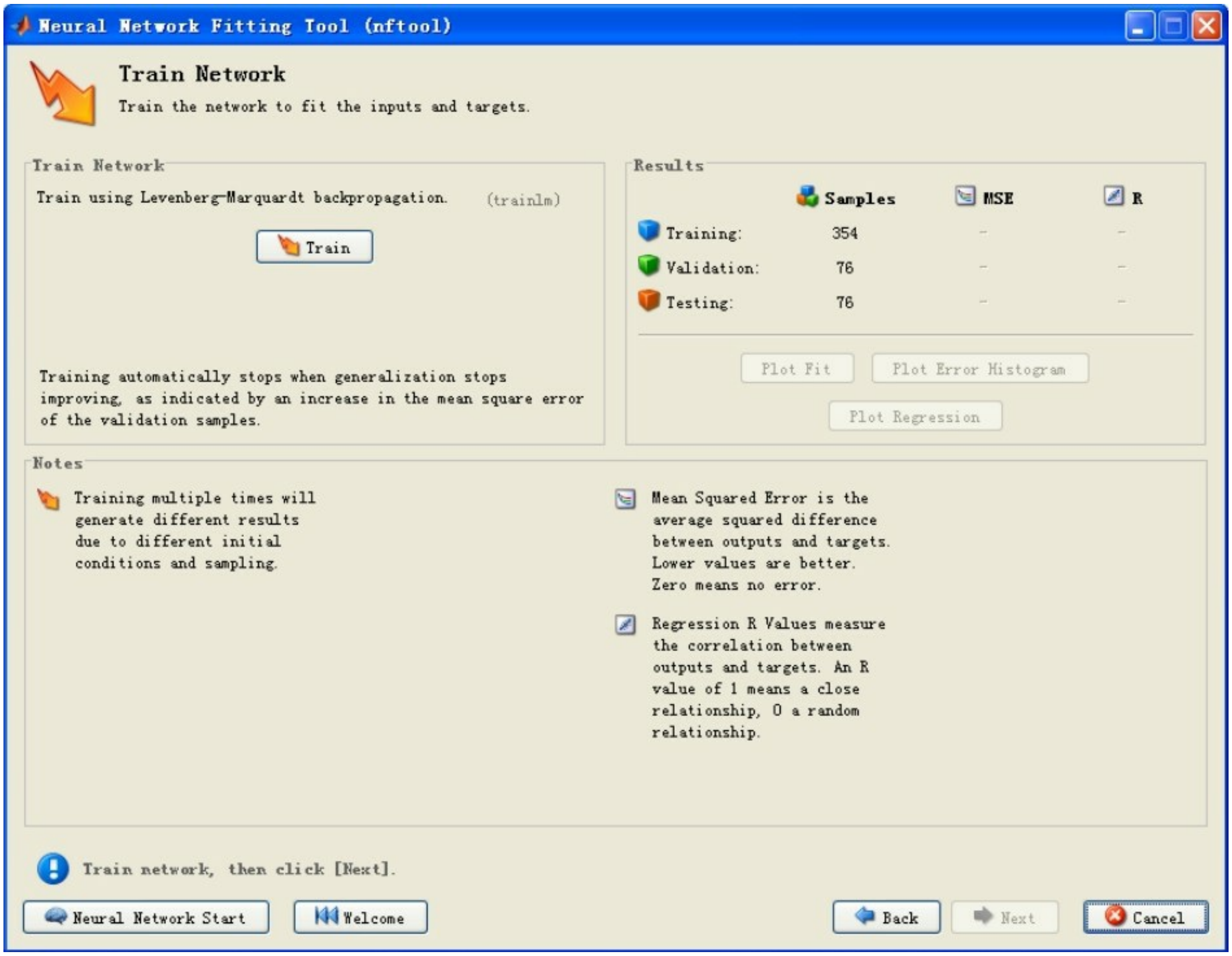


图4-17 神经网络的确定

单击“Train”按钮，选择训练之前确定的神经网络，如图4-18所示。

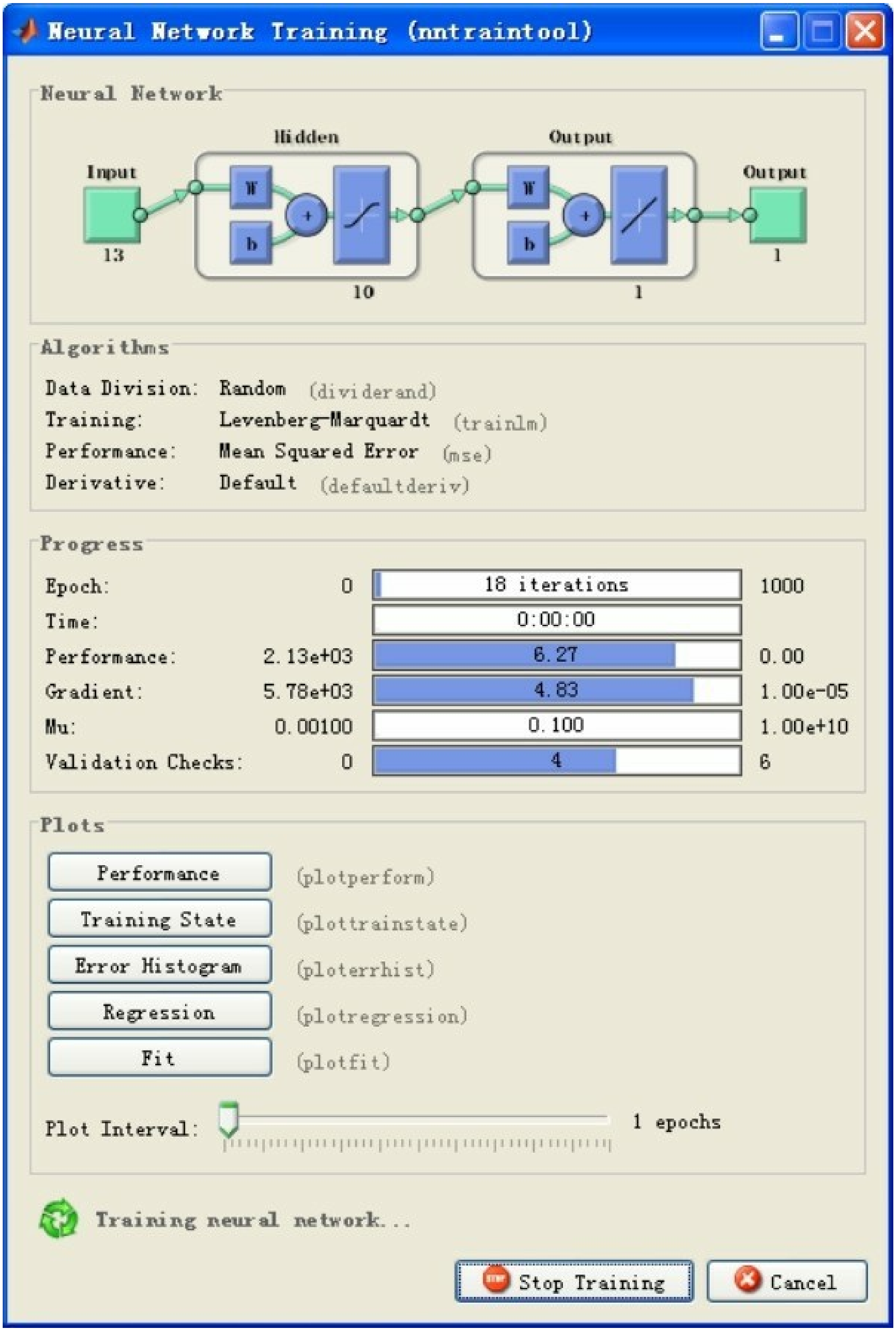

Neural Network Training (nntraintool)
Neural Network
Hidden
Output
Input
Output
W
b
13
10
1
1
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)
Progress
Epoch: 0 18 iterations 1000
Time: 0:00:00
Performance: 2.13e+03 6.27 0.00
Gradient: 5.78e+03 4.83 1.00e-05
Mu: 0.00100 0.100 1.00e+10
Validation Checks: 0 4 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)
Plot Interval: 1 epochs
Training neural network...
Stop Training
Cancel

图4-18 训练神经网络

神经网络训练完成后，可以在“Plots”选项内选择需要看到的图形。

其中，单击“Performance”按钮可以得到MATLAB提供的示例神经网络训练结果图，如图4-19 所示；单击“Training State”按钮可以得到神经网络训练过程参数变化示意图，如图4-20 所示。其他神经网络训练参数，读者可以单击“Error Histogram”、“Regression”和“Fit”按钮查看。

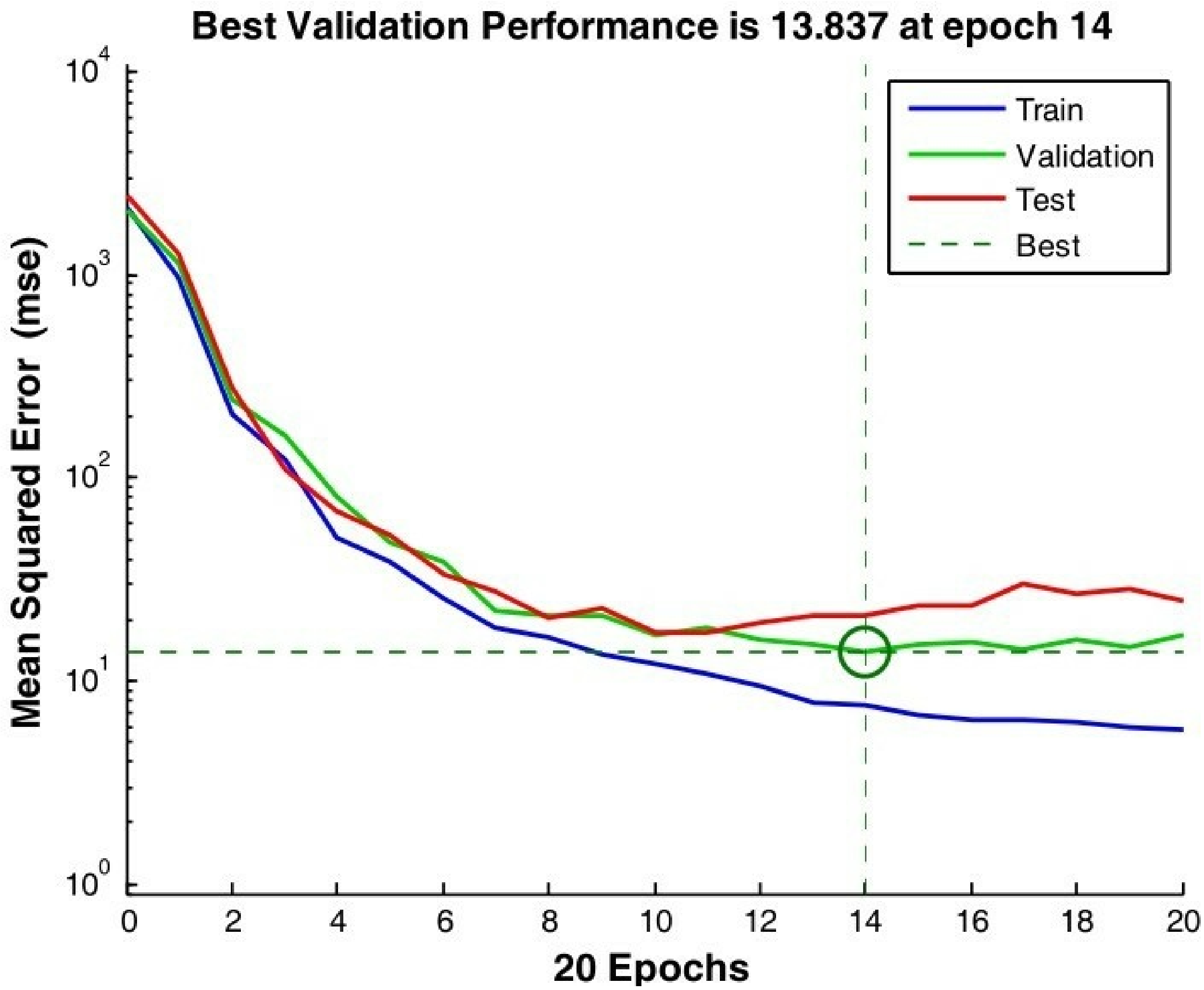


图4-19 训练结果

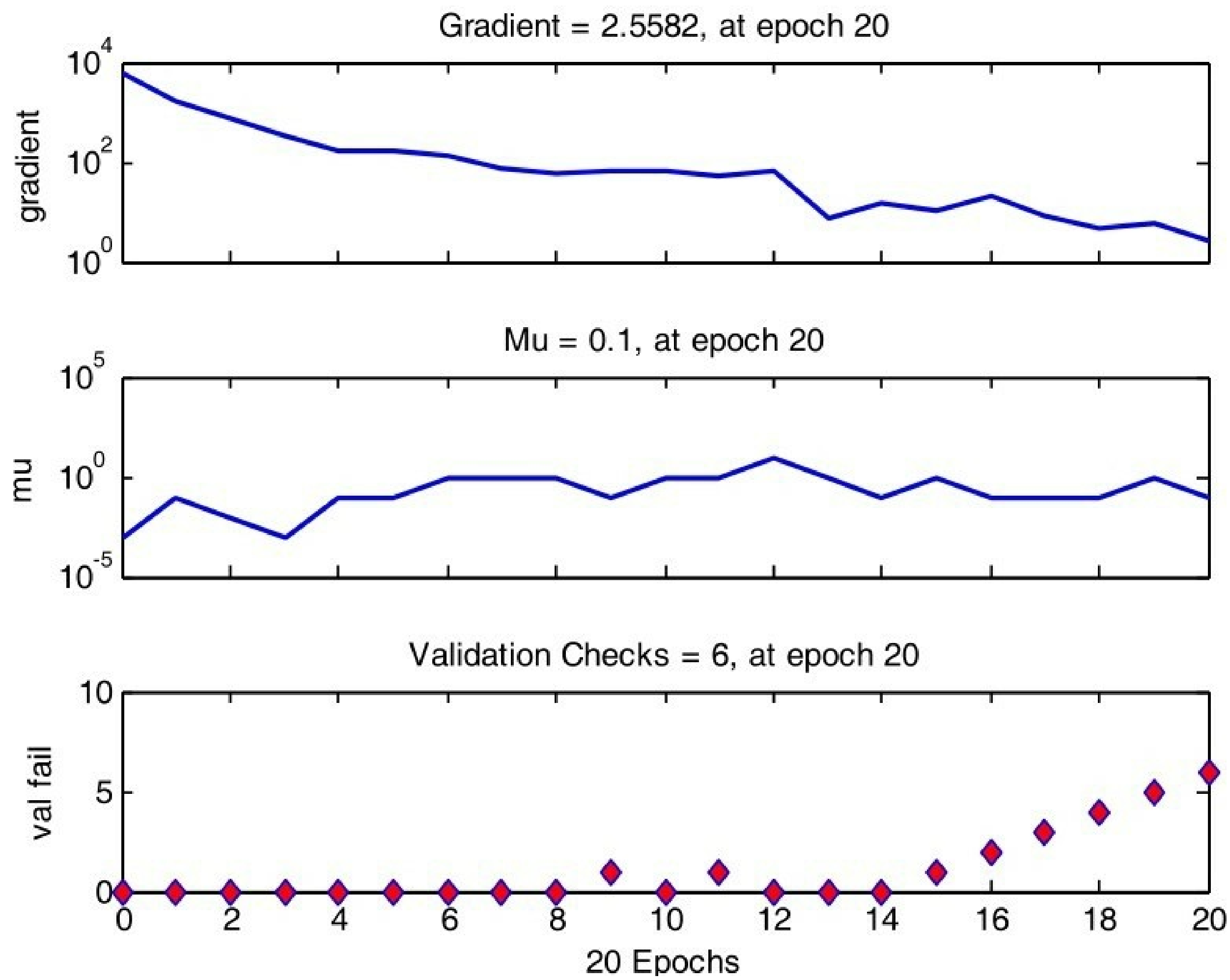


图4-20 训练参数变化曲线

训练完成后，还可以在“神经网络的确定”窗口中连续两次单击“Next”按钮，得到图4-21所示的保存结果界面。

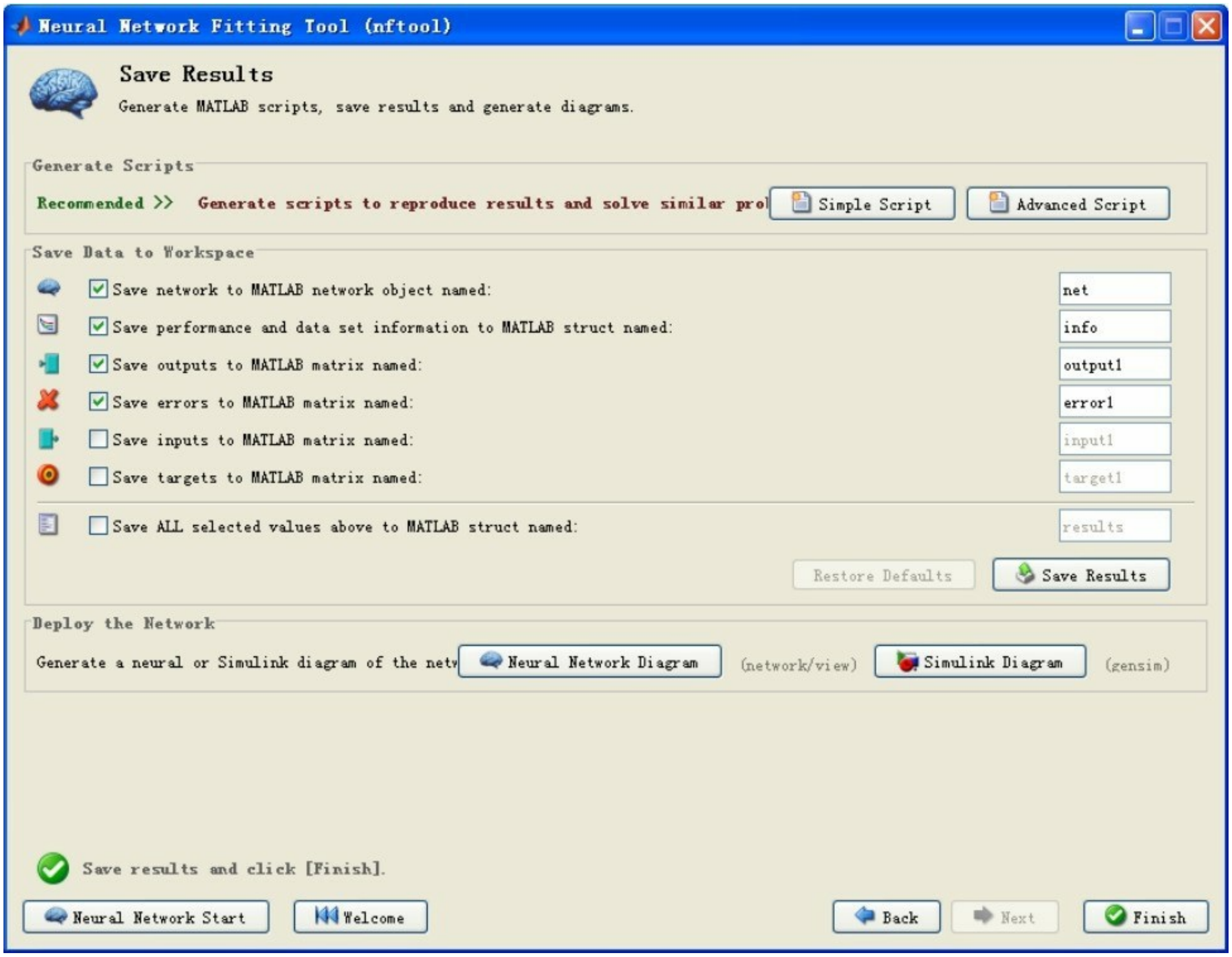


图4-21 保存结果

用户可以在“Save Data to Workspace”选项中选择将哪些数据保存到MATLAB的Workspace中；还可以将神经网络结构保存为图片或者Simulink模型，保存的图片如图4-22所示，保存的Simulink模型如图4-23所示。

MATLAB R2013a 版本里有很多人性化的设计，可以为使用者节省时间，限于篇幅，这里只能说明一部分，更多的还需要读者自己去挖掘。

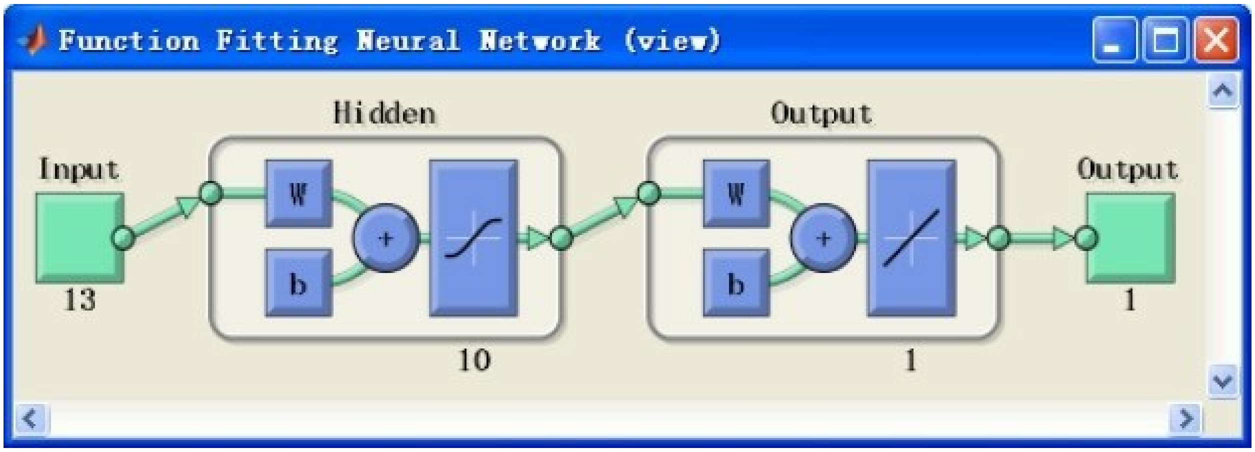


图4-22 神经网络结构保存为图片

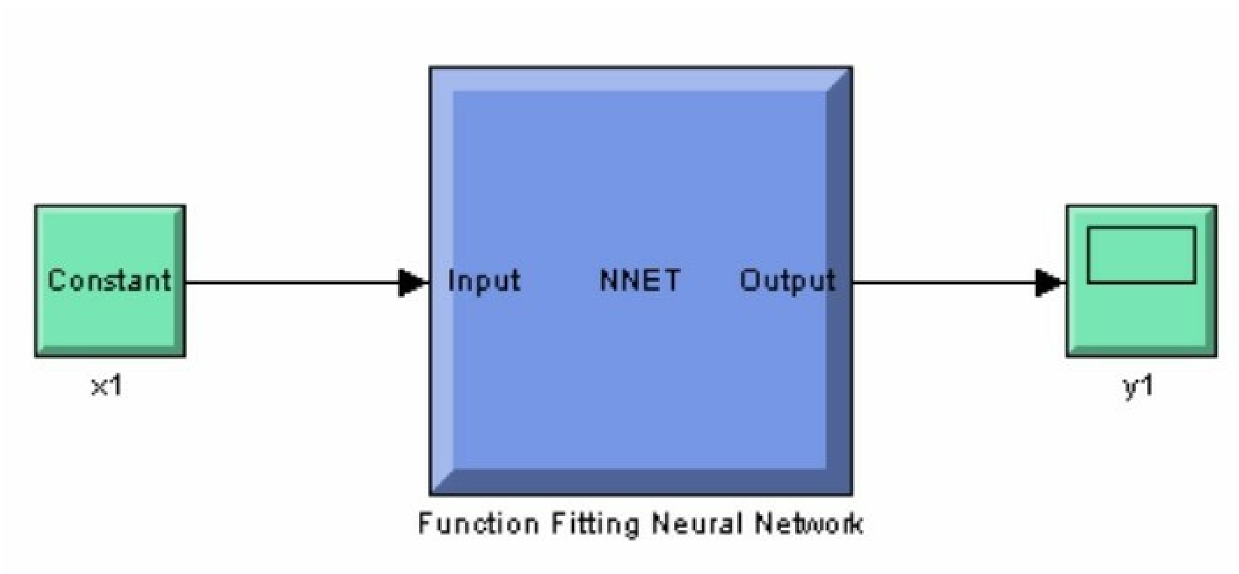


图4-23 神经网络保存为Simulink模型

4.5 本章小结

本章主要介绍了人工神经网络的基础知识，包括神经网络的发展、研究内容、研究方向和发展趋势；介绍了神经网络基础单元——神经元的模型特点和分类等；还介绍了人工神经网络的结构和MATLAB中神经网络工具箱的使用等知识。

第5章 感知器

感知器是模拟人的视觉接受环境信息的过程，由神经冲动进行信息传递的层次型神经网络。感知器的研究首次提出了自组织和自学习的思想，且对所能解决的问题存在着收敛算法，并能从数学上严格证明。

本章将主要介绍感知器的原理、学习算法及其局限性，并对感知器的MATLAB仿真程序设计进行详细说明。

学习目标：

·理解感知器的原理

·了解感知器的局限性

·掌握感知器的仿真程序设计

5.1 感知器原理

感知器是生物神经细胞的简单抽象。神经细胞结构大致可分为：树突、突触、细胞体及轴突。在模拟神经细胞行为过程中，提出了与神经细胞对应的感知器概念，如权量（突触）和传递函数（细胞体）。

感知器在人工神经网络领域中被指为单层人工神经网络，这是为了区别于较复杂的多层感知器。单层感知器可说是最简单的前向人工神经网络形式。作为一种线性分类器，感知器结构简单，但能够学习并解决相当复杂的问题。

本节以单层感知器为例，介绍感知器的原理。

5.1.1 感知器模型

单层感知器是指只有一层处理单元的感知器（如果包括输入层，则为两层）。单层感知器模型如图5-1所示。

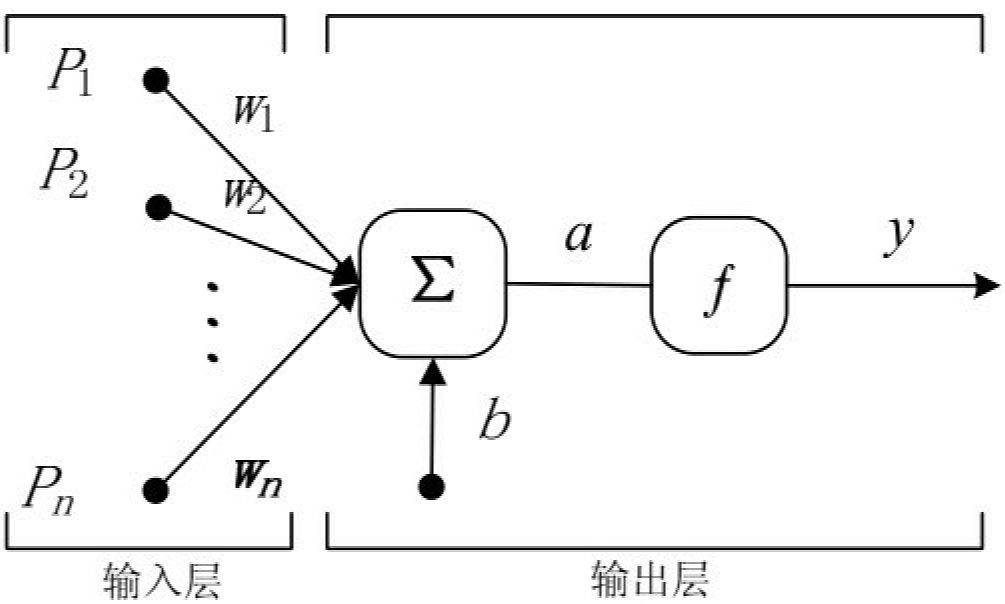


图5-1 单层感知器模型

图5-1中输入层（也称为感知层）有n个神经元节点，这些节点自身无信息处理能力，只负责引入外部信息。

每个节点接收一个输入信号，n个输入信号构成输入列向量P。输出层（也称为处理层）有1个神经元节点，该节点具有信息处理能力。

两层之间的连接权值用权值列向量W_n表示，n个权向量构成单层感知器的权值列向量W。感知器的传递函数为f。2个列向量分别表示为

$$P=(p_1,p_2,\ldots,p_n)^{\mathrm{T}}$$

$$W=(w_1,w_2,\ldots w_n)^{\mathrm{T}}$$

对于处理层中任一节点，由前面介绍的神经元数学模型知，其净输入a为来自输入层各节点的输入加权和

$$a=\sum_{i=1}^{n}p_i w_i+b$$

单层感知器的输出为

$$y=f(a)$$

单层感知器一般选取符号函数作为其传递函数。符号函数的图形如图5-2所示。

如果以两个输入的单层感知器神经元为例，对于选定的权值$w_1=1$和$w_2=-1$，阈值b=1。感知器的输入矢量分类情况如图5-3所示。

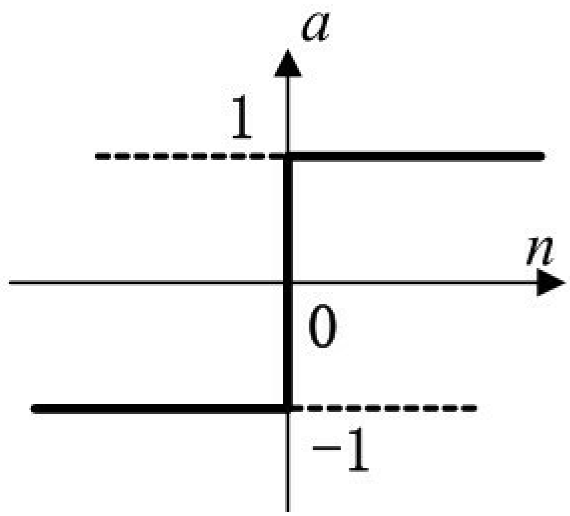


图5-2 符号函数

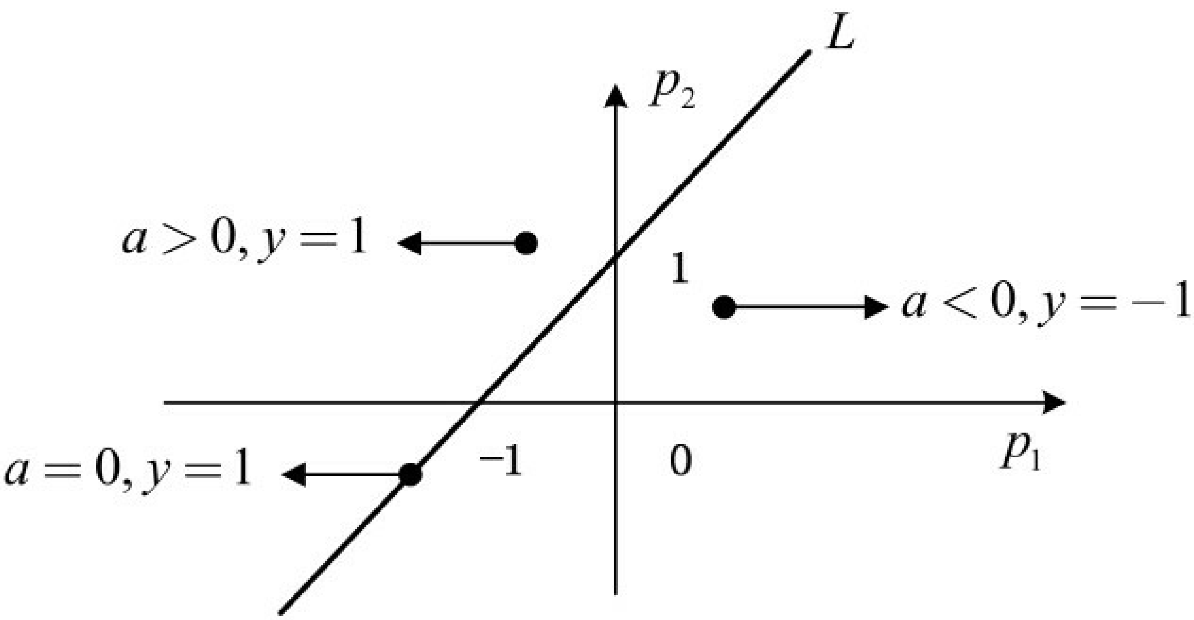


图5-3 输入矢量的分类情况

从图5-3中可以看出，分割线L将两个输入矢量组成的平面分成两个区域：如果输入矢量位于分割线上方，则感知器输入为1；如果输入矢量位于分割线上，则感知器输入为1；如果输入矢量位于分割线下方，则感知器输入为−1。

5.1.2 感知器初始化

当建立好一个感知器后，需要对感知器进行初始化。对感知器的初始化可以采用MATLAB神经网络工具箱中的init()函数。使用init(net)函数可以得到一个已经初始化的神经网络net。

使用 init 函数初始化的神经网络，其网络权值和阈值是按照神经网络初始化函数来进行调整的。网络的初始化函数由NET.initFcn设定，其参数由NET.initParam指定。初始化函数的用法如下。

net=init(net)

MATLAB工具函数中的newp函数可以生成一个感知器网络，其中使用了上述语句对网络net进行初始化。

【例5-1】用init(net)函数设计一个具有2个输入的单个神经元感知器。输入范围是：[0,2]和[-1,1]。在程序最后需要显示设计的感知器权值和阈值。

解：首先使用函数newp生成一个感知器，程序如下所示。

```
net=newp([0 2;-1 1],1);
net.iw{1,1}
net.b{1}
```

上述语句执行结果如下所示。

```
ans =
   0    0
ans =
   0
```

网络的权值和阈值均被设置为0。

如果一个网络被训练过，那么网络的权值和阈值都可能发生变化。这时，可以用　init函数重新初始化该网络的权值和阈值。重新初始化具体程序如下。

```
net=init(net);
net.iw{1,1}
```

net.b{1}

使用上述程序语句之后，神经网络的权值和阈值均被设置为 0，这是神经网络所使用的初始值。

5.1.3 感知器学习规则

神经网络学习规则是用来计算网络新的权值和阈值的算法。感知器就是利用其学习规则来调整网络的权值和阈值，使得网络的输出最终达到目标的期望值。

对应输入向量p，输出向量a，目标矢量为t的感知器，该感知器的学习误差为e，则e=t−a。此时，感知器的权值和阈值修正公式为

$$\Delta w(i,j)=[t(i)-a(i)]\cdot p(j)=e(i)\cdot p(j)$$
$$\Delta b(i)=[t(i)-a(i)]\cdot 1=e(i)\cdot 1$$

其中，i=1,2,...,n表示输出神经节点个数，j=1,2,...,m表示输入神经节点个数。

更新后的权值和阈值分别为

$$w(i,j)=w(i,j)+\Delta w(i,j)$$
$$\Delta b(i)=b(i)+\Delta b(i)$$

感知器的学习规则属于梯度下降法，该法则已被证明：如果能存在收敛点，则算法在有限次的循环迭代后可以收敛到正确的目标矢量。

在MATLAB神经网络工具箱中，感知器的学习规则可以用函数learnp或者learnpn来实现。

5.1.4 感知器训练

神经网络在应用之前必须经过训练，且经过训练决定网络的权值和

阈值。感知器的训练过程如下。

（1）对于给定的输入向量p，计算网络的实际输出a，并与相应的目标向量t进行比较，用得到的误差e，根据学校规则进行权值和阈值的调整。

（2）权值和阈值结果调整后，感知器重新计算网络的在新权值和阈值的作用下，比较输入值和目标值，重复进行权值和阈值的调整过程。

（3）当网络的输出与目标值相等或者训练的次数达到预先设定的最大值时，训练结束。

经过训练的感知器网络需要验证其是否能够达到要求，即感知器网络是否训练成功。如果感知器网络训练成功，那么训练后的网络对应被训练的每一组输入向量都能产生一组对应的期望输出。

如果在给定的最大训练次数和输入向量作用下，网络没有能够完成训练（即满足 a=t的要求），那么可以增加训练的次数，继续训练网络。

如果在经过足够多次的训练后，网络仍然达不到要求，那么需要分析一下，感知器是否适合解决目前的问题。

感知器并不是适用于任何分类问题。在MATLAB的神经网络工具箱内，感知器的训练函数为adaptwb和trainwb。

5.2 感知器的局限性

感知器的局限性在于以下4点。

（1）感知器的传递函数一般采用阈值函数，所以输出值只有两种（0和1，或者−1和1）。

（2）单层感知器网络只能用于解决线性可分的分类问题，而对线性不可分的分类问题无能为力。如何确定输入向量是否线性可分，至今还没有有效的解决办法，尤其当输入向量增多时，更加难以确定。

（3）感知器学习算法只用于单层感知器网络，所以一般感知器网络都是单层的。

（4）当感知器的所有输入样本中存在奇异的样本，即该样本向量同其他所有的样本向量比较起来特别大或特别小时，网络训练所花费的时间将很长。例如，当输入和目标向量分别为如下内容时。

P=[-0.2 -0.3 0.1 0.2;0.1 -0.3 -0.2 200];

T=[1 1 1 0 1];

由于输入的最后一个数远远大于其他输入数据，这必然导致训练的困难。解决此问题的方法是采用标准化感知器学习规则。在神经网络工具箱中，实现标准化感知器学习规则的函数是learnpn。

5.3 感知器工具箱的函数

MATLAB的神经网络工具箱中提供了大量的感知器函数，下面对这些函数的功能、调用格式、使用方法及注意事项做详细说明。

常用感知器函数见表5-1。

表5-1 感知器常用函数

函数类别	名称	用　　途
感知器创建函数	newp	创建一个感知器网络
显示函数	plotpc	在感知器向量中绘制分界线
	plotpv	绘制感知器的输入向量和目标向量
性能函数	mae	平均绝对误差函数

1.感知器创建函数

通过感知器生成函数可以创建一个感知器，并且可以对感知器进行初始化、训练和仿真等。

感知器生成函数newp用于创建一个感知器网络，调用格式如下所示。

net=newp

net=newp(pr,s,tf,lf)

其中，net表示生成的包含返回参数的感知器网络。

pr表示一个R×2的矩阵（R为输入向量的个数），由输入向量的最大值和最小值组成。

s表示神经元的个数。

tf表示感知器的传递函数，默认的函数为hardlim，其他的传递函数还包括hardlims。

lf表示感知器的学习函数，默认的函数为learnp，其他的学习函数还包括learnpn。

2.显示函数

（1）分界线绘制函数plotpc。

该函数的作用是绘制感知器向量图中的分界线，其调用格式如下所示。

plotpc(W,B) %返回的是绘制分界线的控制权

plotpc(W,B,H) %包含从前一次调用中返回的句柄。它在画新分界线之前，删除旧线。

其中，W表示一个S×R维的加权矩阵（R不大于3）。

B表示S×1维的阈值向量。

H表示最后一次绘制分界线的句柄。

提示：该函数不改变现有的坐标轴，一般在plotpv函数之后使用。

（2）输入/目标向量绘制函数plotpv。

该函数的作用是绘制感知器的输入向量和目标向量，调用格式如下。

plotpv(P,T) %以T为标尺，绘制P的列向量

plotpv(P,T,V) %在V的范围中，以T为标尺，绘制P的列向量

其中，P表示n个2或3维的样本，是一个2×n维或3×n维的矩阵。

T表示一个样本点的类别，是一个n维的向量。

V表示设置坐标值范围的一个向量，这个向量由输入向量的最大值和最小值组成。

利用plotpv函数可以在坐标图中绘出给定的样本点及其类别，不同的类别使用不同的符号。

例如，如果T只含一元向量，则目标为0的输入向量在坐标图中用“o”表示；目标为1的输入向量在坐标图中用“+”表示。如果T含二元向量，则输入向量在坐标图中采用的符号分别为：[0 0]用“o”表示；[0 1]用“+”表示；[1 0]用“*”表示；[1 1]用“×”表示。

【例5-2】如果给出某感知器的输入变量P和目标变量T，绘制其曲

线。如果给定感知器的权值和阈值，绘制其分界线。

解：MATLAB代码如下。

```
P = [0 0 1 1; 0 1 0 1];
T = [0 0 0 1];
plotpv(P,T)      %绘制输入向量和目标向量
```

绘制的输入和目标向量图形如图5-4所示。

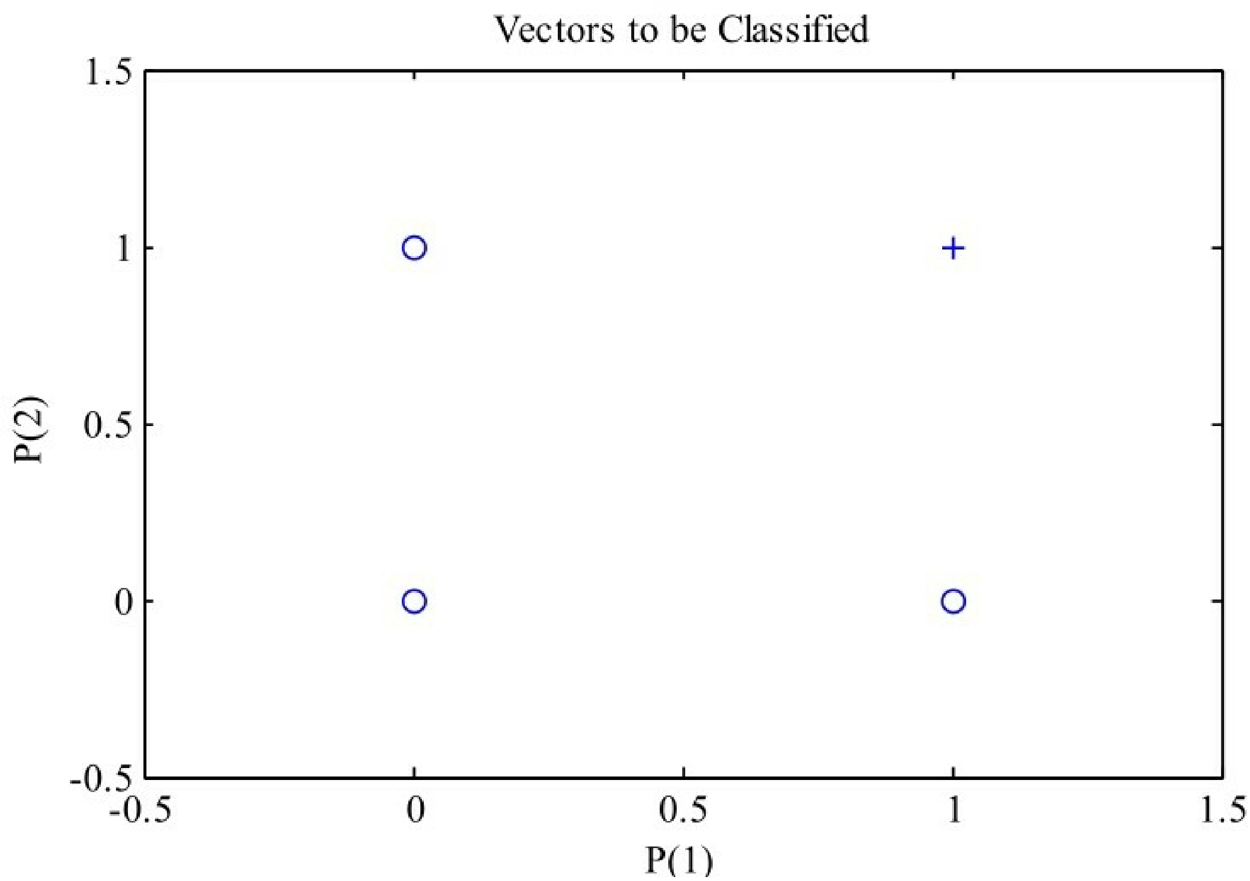


图5-4 输入和目标向量图形

下面的代码创建在P值范围内的感知器，并将其他值赋给感知器的权值和阈值，绘制其分界线。

```
net= newp(minmax(P),1);   %根据输入p，创建一个感知器网络
net.iw{1,1}= [-0.8 -0.5];   %设定权值
net.b{1}= 1;      %设定阈值
plotpc(net.iw{1,1},net.b{1}) %绘制分界线
```

其中，minmax为数学函数，minmax(P)表示取向量P范围内的最大和最小值。

绘制的分界线如图5-5所示。

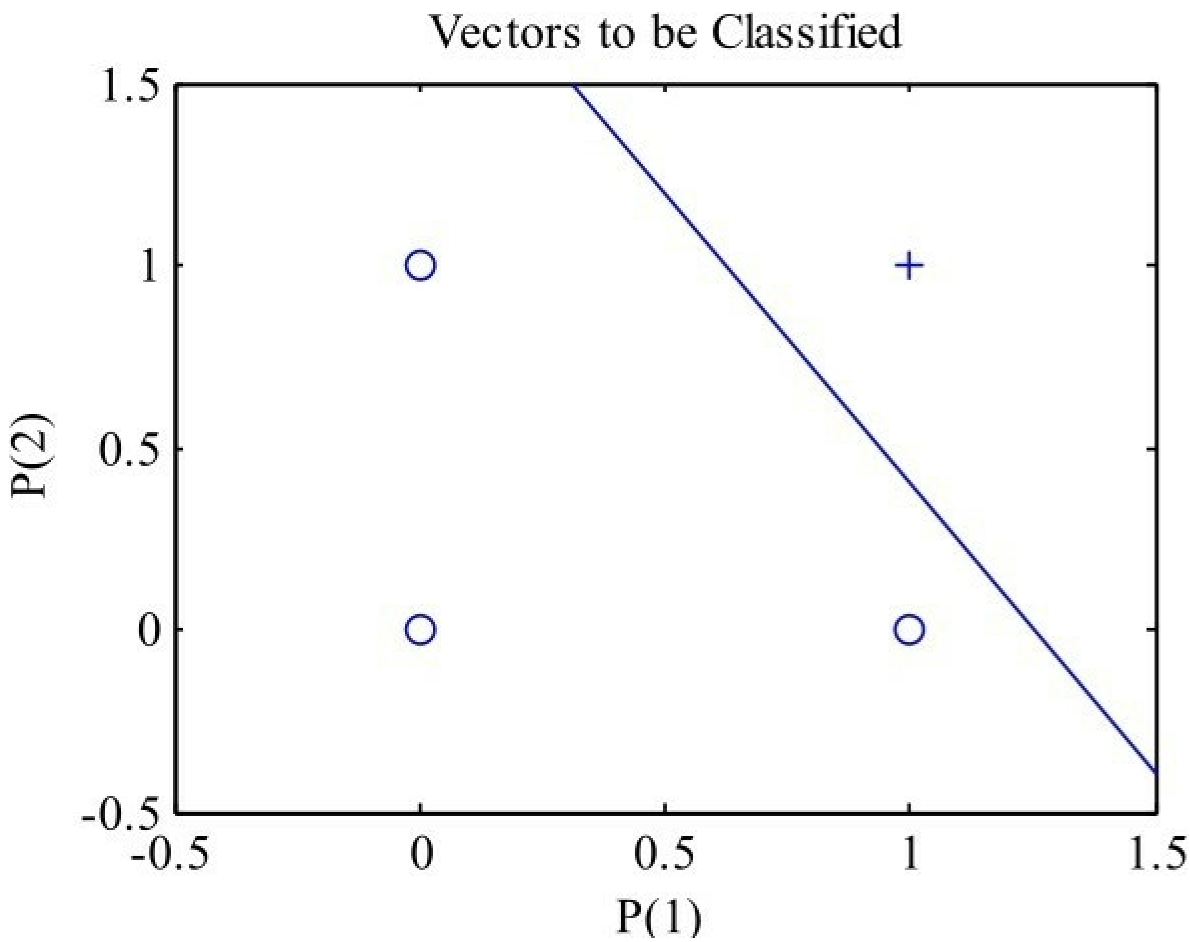


图5-5 绘制的分界线

（3）性能函数mae。

该函数以平均绝对误差为基础，确定感知器性能的函数。其调用格式如下所示。

```
perf = mae(E,Y,X,FP)
dPerf_dy = mae('dy',E,Y,X,perf,FP)
dPerf_dx = mae('dx',E,Y,X,perf,FP)
info = mae('code')
```

其中，perf表示函数的返回值，为平均绝对误差和。

E为误差矩阵或向量（网络的目标向量和输出向量之差）。

Y为阈值向量（可忽略）。

X为所有权值（可忽略）。

FP为性能参数（可忽略）。

mae(code)可根据code的不同，返回不同信息，code包括以下内容。

·deriv：返回导数函数的名称。

·name：返回函数全称。

·pnames：返回训练函数的名称。

·pdefaults：返回默认的训练参数。

【例5-3】建立一个输入向量为从−10到10的感知器，并使用性能函数计算其平均绝对误差。

解：MATLAB代码如下所示。

```
net= newp([-1010],1);    %建立一个感知器
p= [-10 -50 510];    %设定输入向量
t= [0 01 11];        %设定目标向量
y= sim(net,p)        %计算感知器的输出
e= t-y            %计算目标向量与输出向量之差
perf=mae(e)        %计算平均绝对误差
```

输出结果如下所示。

```
y =
   1    1    1    1    1
e =
   -1        -1    0    0    0
perf =
   0.4000
```

【例5-4】感知器最重要的也是最实用的功能是对输入向量进行分类。尝试建立一个感知器模型，实现电路中“或”门的功能，从而实现对

输入的分类。其中，“或”门的输入/输出关系见表5-2。

表5-2 “或”门的输入/输出关系

输　入	输　出
00	0
01	1
10	1
11	1

解：MATLAB代码如下。

```
P=[0 1 11;0 101];    %输入向量
T=[0 11 1];        %输出向量
net=newp(minmax(P),1);    %建立感知器
Y=sim(net,P)        %仿真
net.trainParam.epochs=50; %设置最大训练步数
net=train(net,P,T);
Y=sim(net,P)
perf=mae(Y-T)
```

代码运行结果如下所示。

```
Y =
   1    1    1    1
Y =
   0    1    1    1
perf =
   0
```

感知器训练对话框如图5-6所示。

从代码运行结果可以看出，第1次计算得到的输出与目标变量不一致，经过2次训练，感知器的输出已经和目标向量一致了。在图5-6中，Progress选项中的Epoch参数为2时，感知器停止训练，这表示经过2次训练，感知器的输出向量与目标向量达到一致。

感知器训练过程的误差曲线如图5-7所示。从该图中也可以看出，经过2次训练，感知器输出达到目标值。

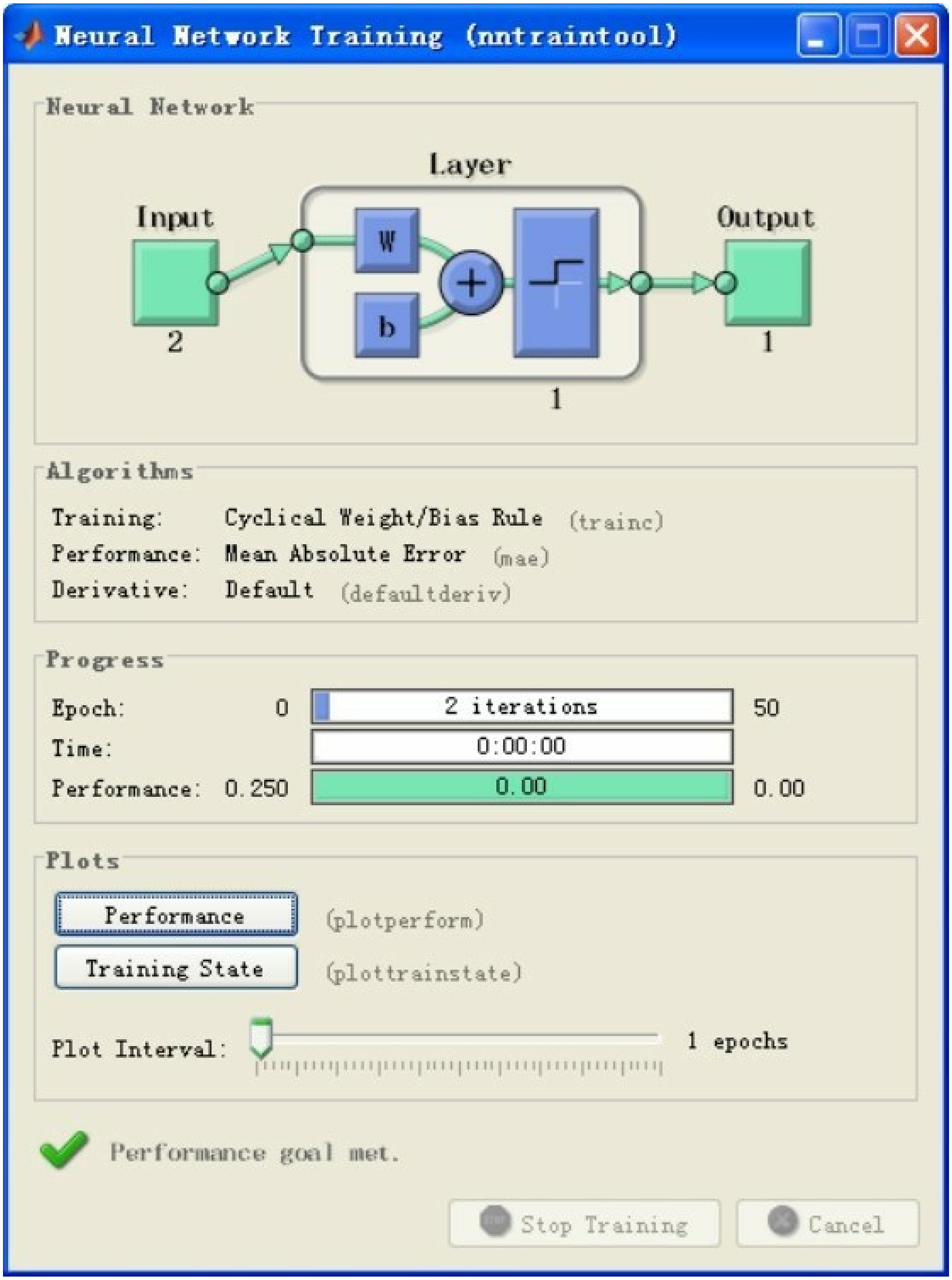


图5-6 感知器训练对话框

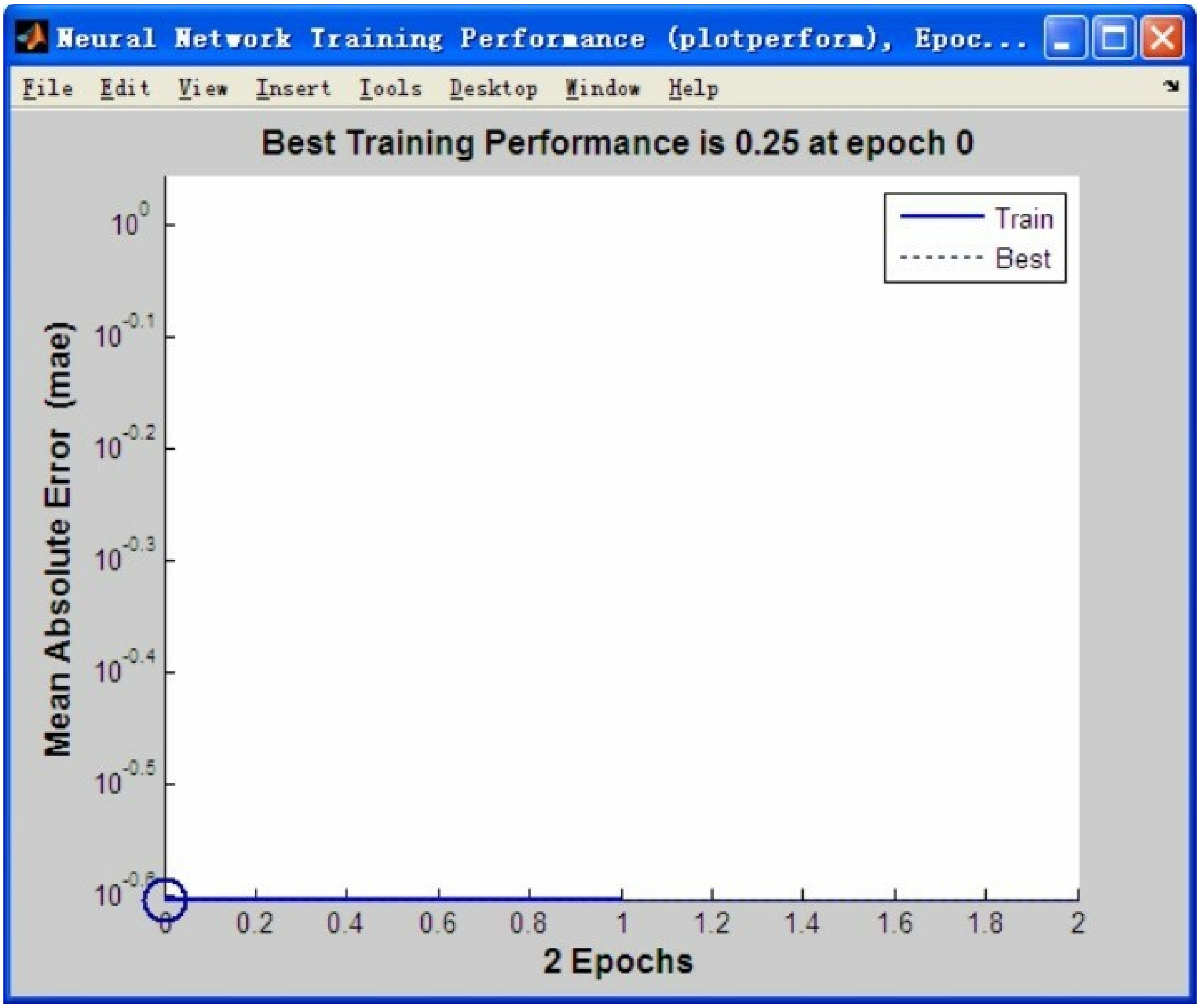


图5-7 训练过程的误差曲线

利用 trainc 函数对感知器进行训练，训练的结果非常理想，训练后的网络成功实现了“或”功能。利用性能函数mae，计算网络的性能，结果为0，这说明了网络的性能非常好。

上例创建的感知器只有一个神经元，这符合了神经网络的设计原则：在设计神经网络时，在同样可以达到目标值的情况下，尽量采用简单的网络结构。神经网络的结构越简单，其计算负担越轻，运行速度越快。

感知器的传递函数和学习函数都采用默认值，分别为haedlim和learnp。传递函数的选取是因为感知器的输出为0-1的二值结构，只有采

用hardlim才满足要求。学习函数的选取是因为输入向量中不存在奇异值，元素之间的距离也比较小，因此，采用learnp就足够了。

利用自适应函数adapt同样可以达到训练的效果。采用以下代码对感知器进行训练。

```
P=[0 1 1 1;0 1 0 1];
T=[0 1 1 1];
net=newp(minmax(P),1);
Y=sim(net,P)
e=Y-T;
while mae(e)
   [net,Y,e]=adapt(net,P,T);
end
Y=sim(net,P)
```

代码运行结果如下所示。

```
Y =
   1    1    1    1
Y =
   0    1    1    1
```

训练过程中采用mae函数来得到感知器误差，以此作为是否停止训练的标准。此时的感知器输出和前面的一致，因此，采用这种方式对网络进行训练是成功的。

【例5-5】要求建立一个感知器，对一组存在奇异值的输入向量P进行分类。

解：网络的输入向量P和目标向量T分别如下。

```
P=[0 0.5 1.1 21;0 0.7 1.3 70];
T=[0 1 0 1];
```

其中，T表明了输入向量的分类情况。

输入向量的分布情况如图 5-8 所示。其中的一个坐标点与其他点的距离都很大，而另外3个点相对比较集中，因此，输入向量中存在奇异值。

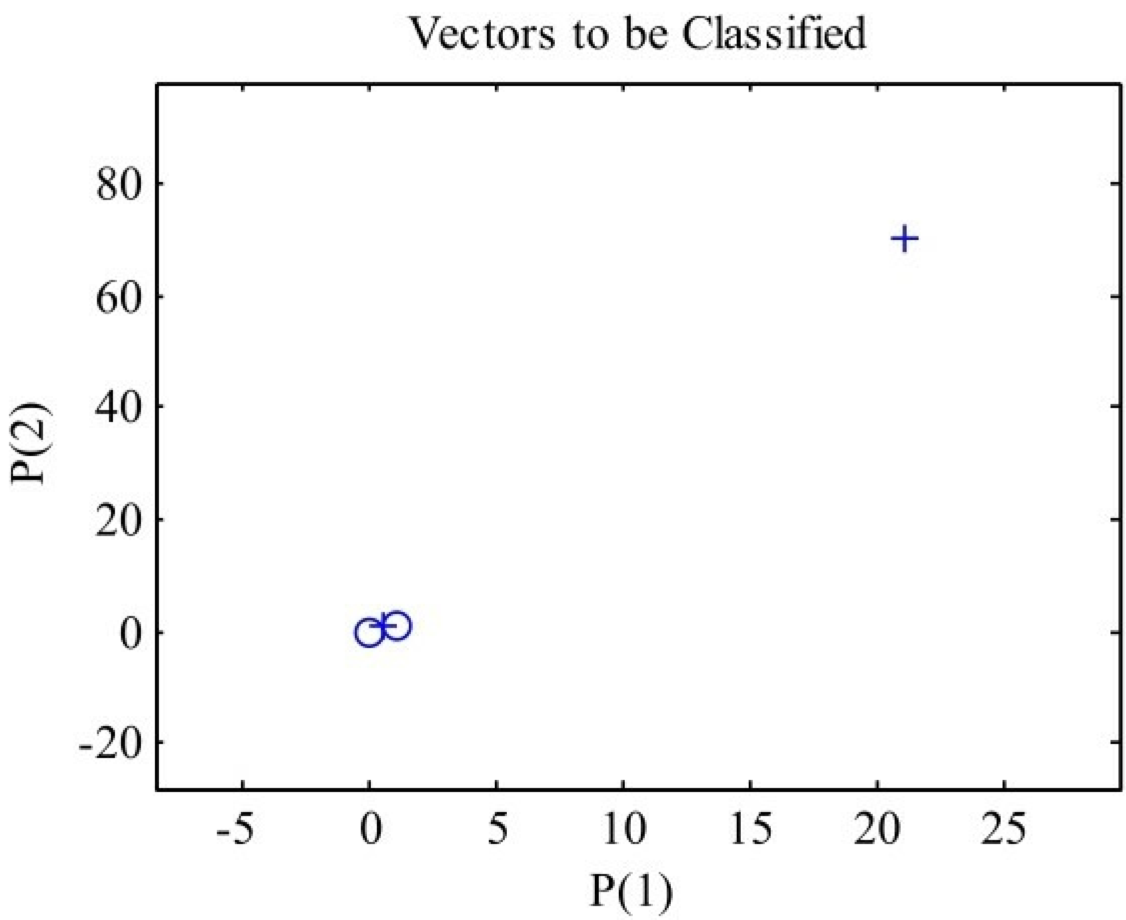


图5-8 输入向量的分布情况

MATLAB代码如下所示。

```
P=[0 0.5 1.1 21;0 0.7 1.3 70];
T=[0 1 0 1];
net=newp(minmax(P),1);
net.trainParam.epochs=500;
net=train(net,P,T);
figure;
plotpv(P,T);
```

plotpc(net.iw{1},net.b{1});

训练过程中的误差曲线如图 5-9 所示。从图中可以看出，网络的训练过程中出现了振荡，而且训练次数也比较多。

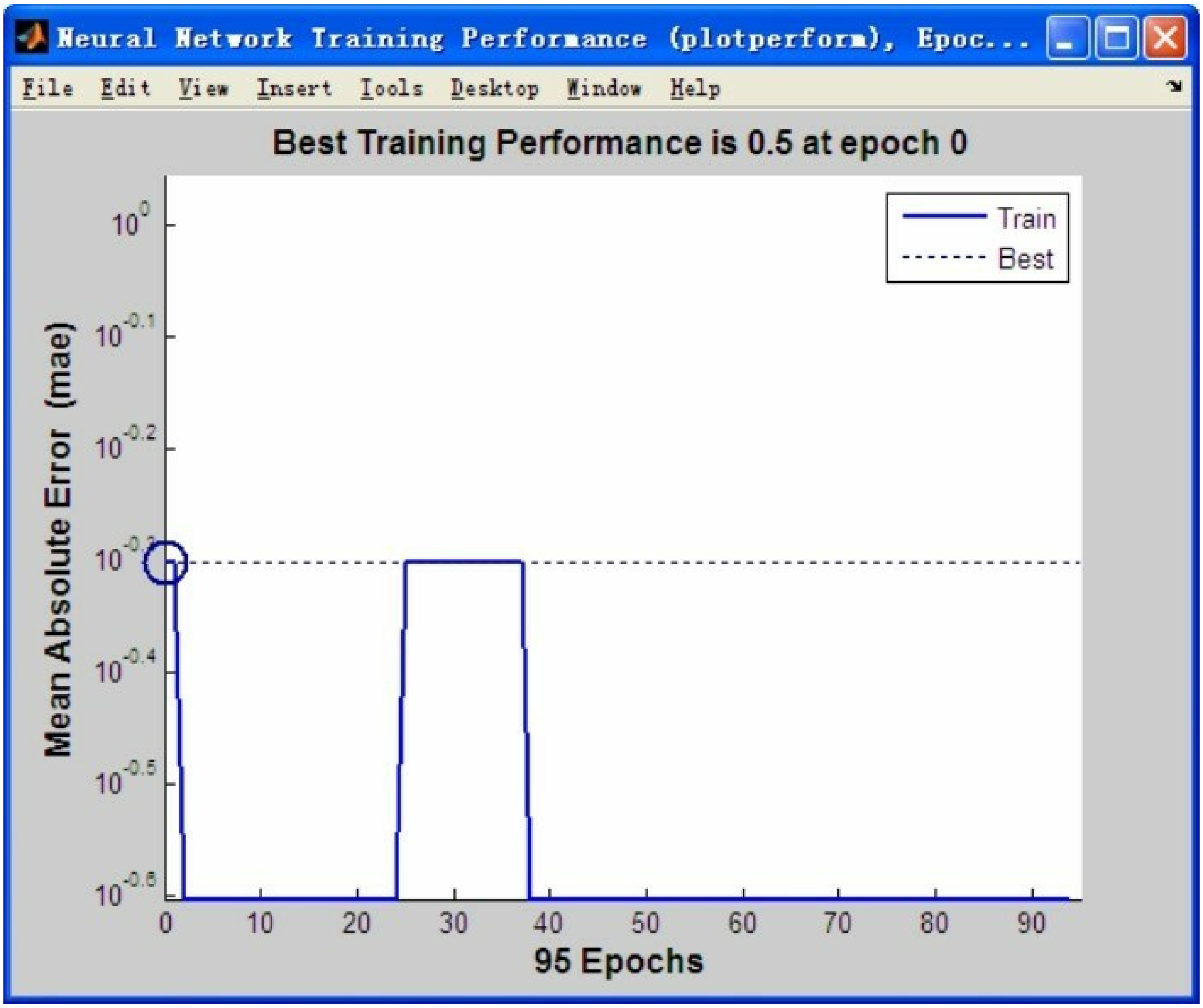


图5-9 训练过程中的误差曲线

训练得到的分界线如图5-10所示。

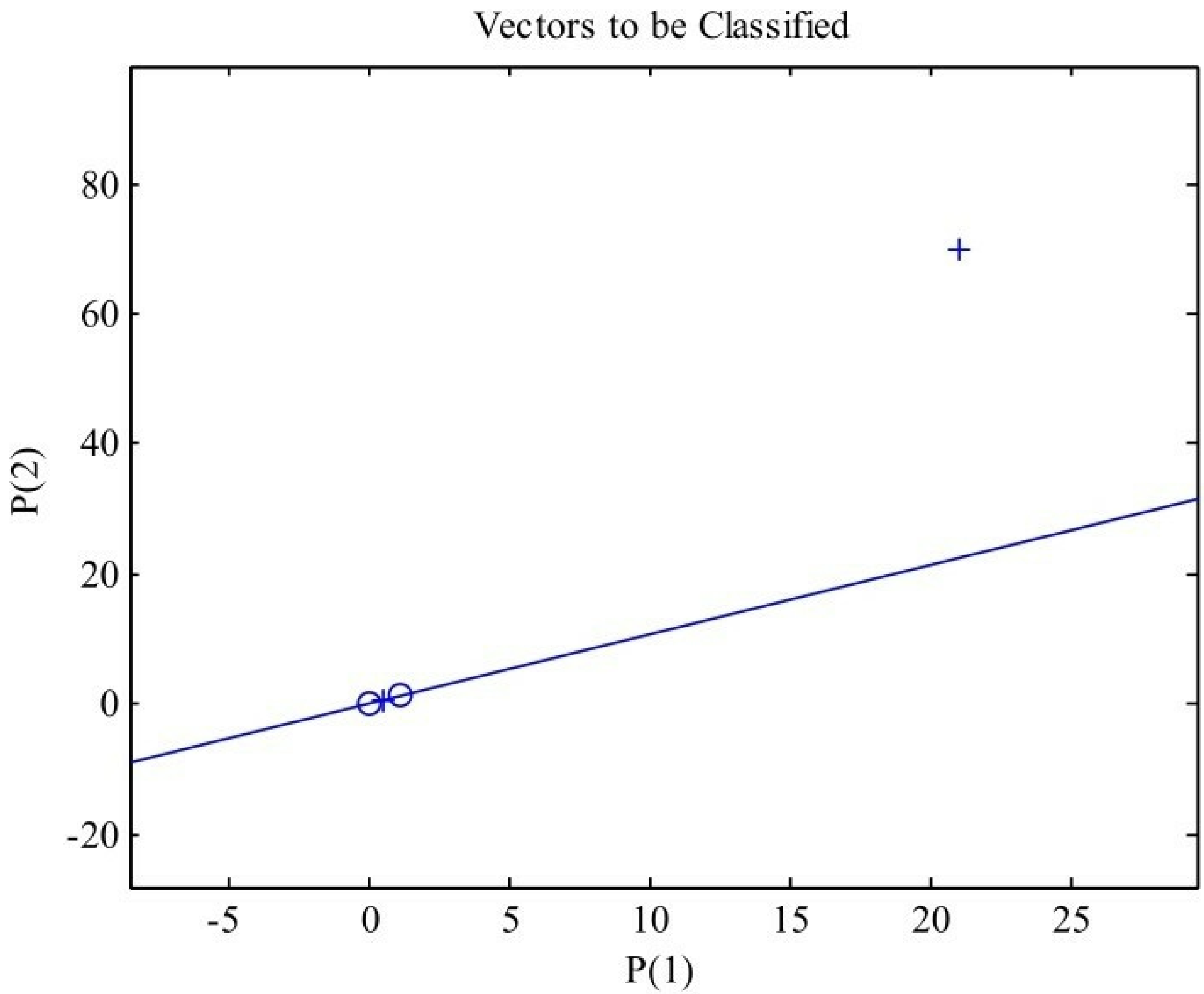


图5-10 训练得到的分界线

由于奇异值的存在，样本点相对比较集中，因此无法看出网络的分类性能。接下来将局部放大，检查网络的分类性能。局部放大的代码如下所示。

```
figure;
plotpv(P,T);
plotpc(net.iw{1},net.b{1});
axis([-2 2 -2 2])
```

运行以上代码得到局部放大后的分界线，如图5-11所示。

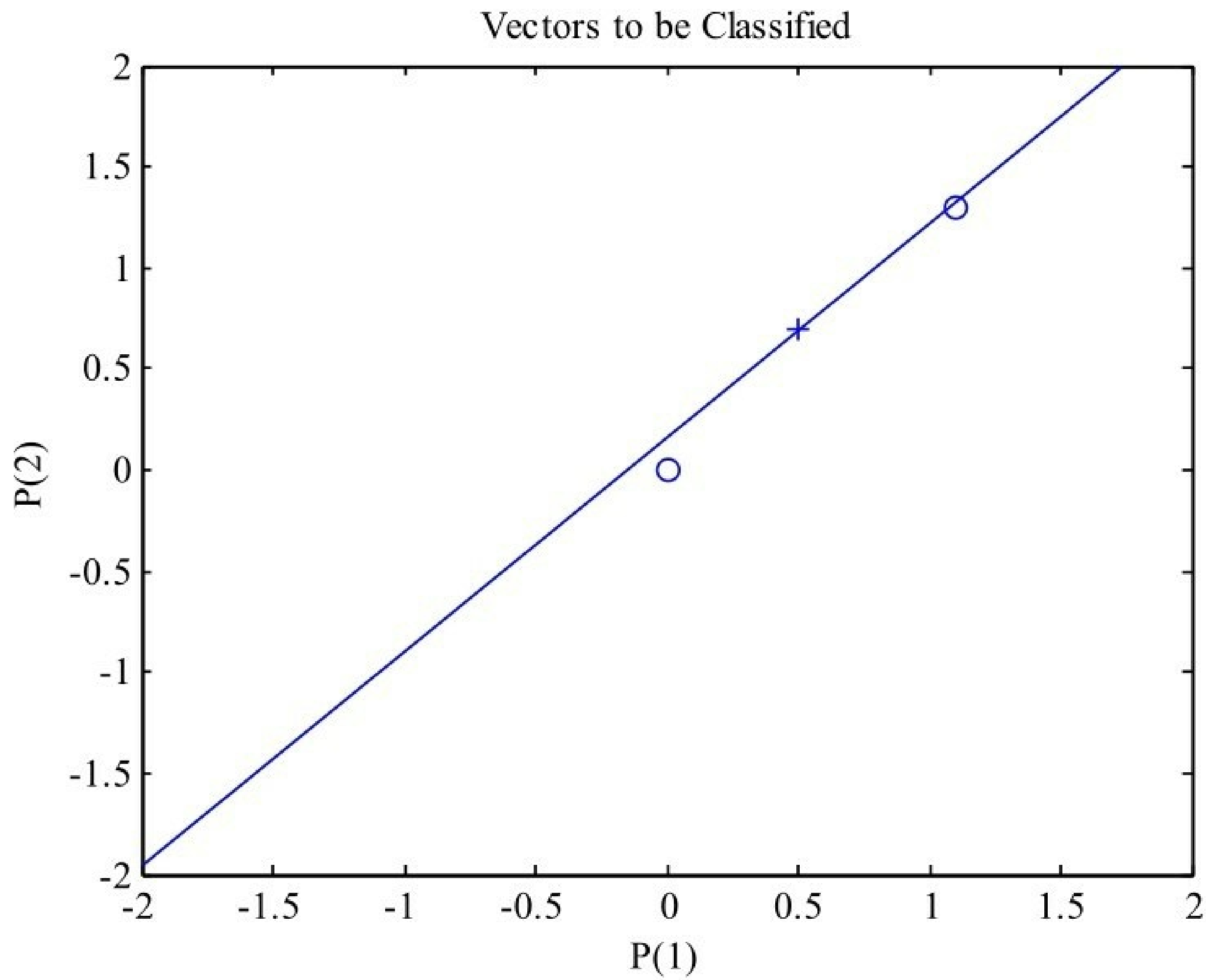


图5-11 局部放大后的分界线

两类样本点中分别有一个位于分界线上，由此可见，输入样本中的奇异点对于网络的分类性能影响很大。

以上代码创建感知器的学习函数为learnp，接下来利用标准训练函数learnpn作为网络的学习函数。除了感知器创建代码有区别之外，其余部分均一致。MATLAB代码如下所示。

```
net=newp(minmax(P),1,'hardlim','learnpn');
```

网络的训练误差曲线如图5-12所示，由图可见，感知器经过189次学习后性能为0，相对于图5-9中的95次，网络训练时间增加了一倍。

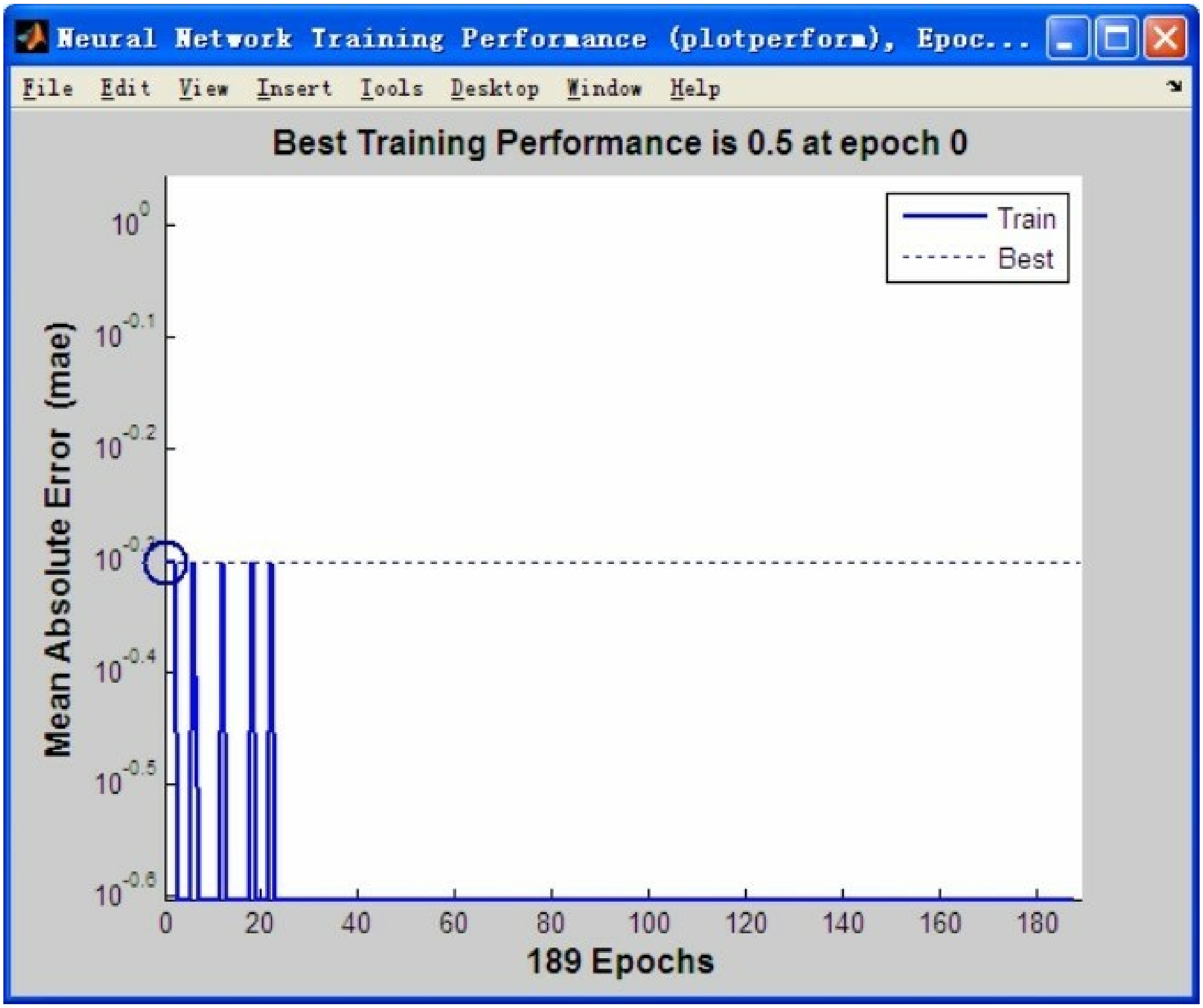


图5-12 网络的训练误差曲线

此时感知器得到局部放大后的分界如图5-13所示，图5-13的分界线没有图5-11中的分界线准确。

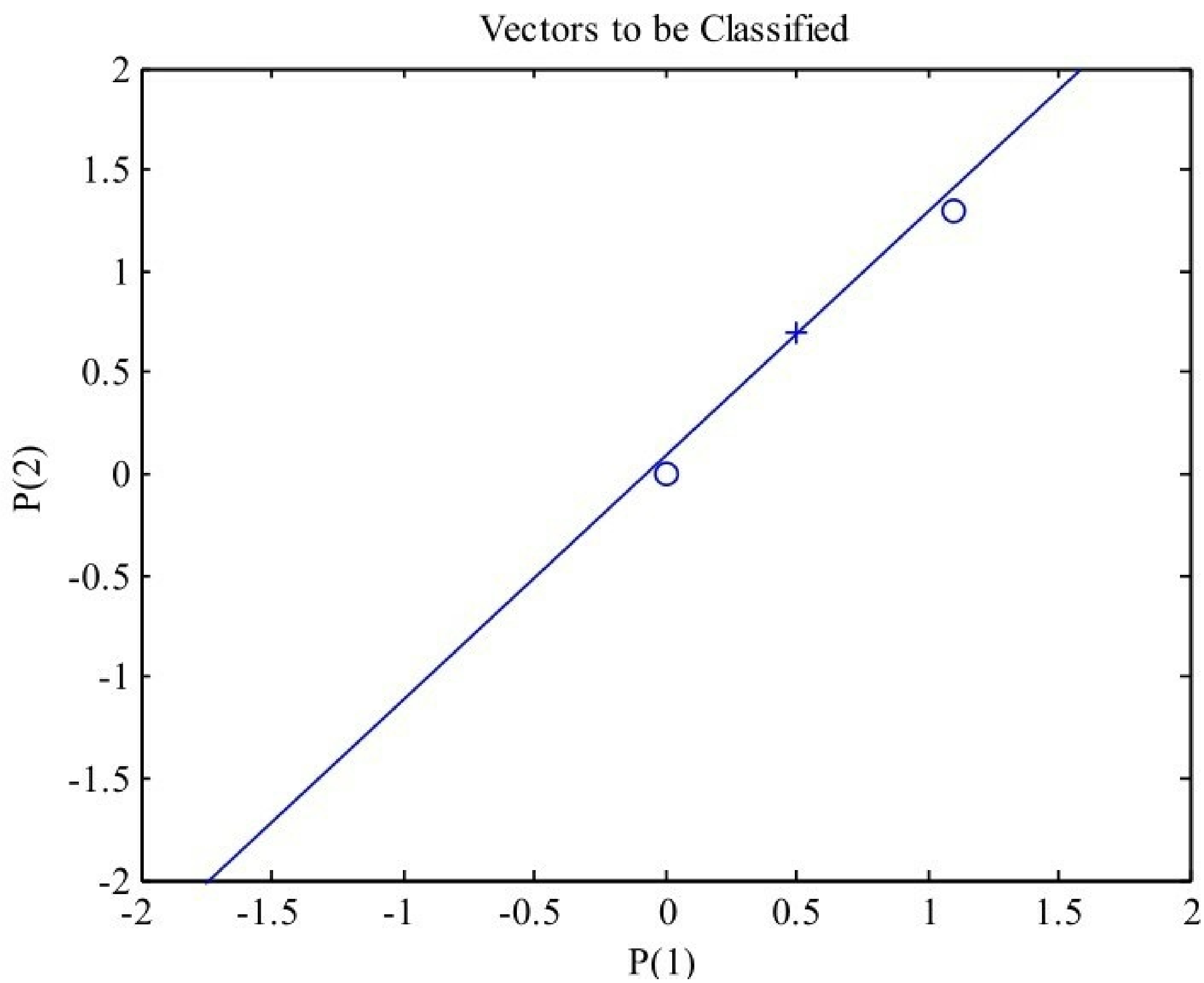


图5-13 局部放大后的分界线

5.4 感知器的MATLAB仿真程序设计

根据前面几节关于感知器模型、函数等的介绍，本节详细介绍单层感知器和多层感知器的MATLAB仿真方法。

5.4.1 单层感知器MATLAB仿真程序设计

单层感知器的MATLAB仿真程序设计主要包括以下3个方面。

（1）创建感知器。

首先根据需要解决的问题，确定输入向量的取值范围和维数。其次，确定网络层神经元的个数、传递函数和学习函数等。最后，以单层感知器的创建函数 newp 创建感知器神经网络。

（2）训练创建的网络。

建立训练样本集，确定每个样本的输入向量和目标向量。调用函数train对创建好的感知器神经网络进行训练，并根据训练的情况决定是否调整训练参数，以得到满足误差性能指标的感知器神经网络，并存储训练好的感知器神经网络参数。

（3）对训练好的感知器进行仿真。

建立测试样本集，加载训练后的网络，调用函数 sim，以测试样本集进行仿真，检测训练好后的网络性能。

下面用经典的例题来说明单层感知器的MATLAB仿真程序设计。

【例 5-6】设计一个单层单输出感知器神经网络，进行二值化图像卡片上数字的奇偶分类。

解：图5-14中给出了数字0和1的二值化图像卡片，每一个图像卡片可以分成5×3的矩形方框。如果有数字的笔画划过的每个小方块（即在

小方块内二值图像元素的值至少有一个不为0）记为1，没有数字的笔画划过的小方块记为0。那么图像卡片上所有小方块表达了有0和1二值组成的一个向量，该向量可以作为感知器神经网络的输入向量。

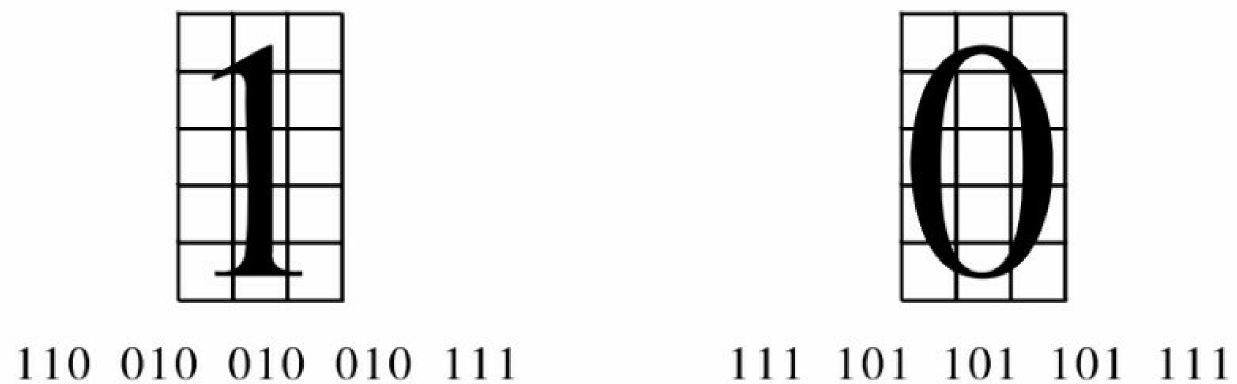


图5-14 图像数字卡片构成的向量示意图

按照上述方法，由数字0～5构成的输入向量见表5-3。

表5-3 数字0～5构成的输入向量

数字	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}
0	1	1	1	1	0	1	1	0	1	1	0	1	1	1	1
1	0	1	0	0	1	0	0	1	1	0	1	0	1	1	1
2	1	1	1	0	0	1	0	1	1	1	1	0	1	1	1
3	1	1	1	1	1	1	0	1	0	0	0	1	1	1	0
4	0	1	0	1	1	0	0	1	0	1	1	1	0	1	0
5	1	1	1	0	1	0	1	1	1	0	0	1	1	1	1

当然，相同数字的形状不同、位置不同、笔画的粗细不同等，都会引起相应的输入向量的变化。

如果网络输出在图像卡片上的数字为奇数时，设计的单层感知器的输出为0，否则为1，那么可以根据上述规则完成其奇偶分类。

结合以上分析，根据例题所要求的感知器神经网络设计的基本结构如下。

·感知器有1 个输入向量，包括15 个元素，这些元素与图像卡片上的15 个小方块的值对应，输入元素的取值范围为[0 1]。

·设计的感知器为单层、单神经元感知器神经网络。

·输入是一个二值向量0　或1，它的两种不同取值分别表示分类结果的奇偶情况，所以神经元的传输函数可以取为logsig函数。

最终得到感知器神经网络的结构示意图如图5-15所示。

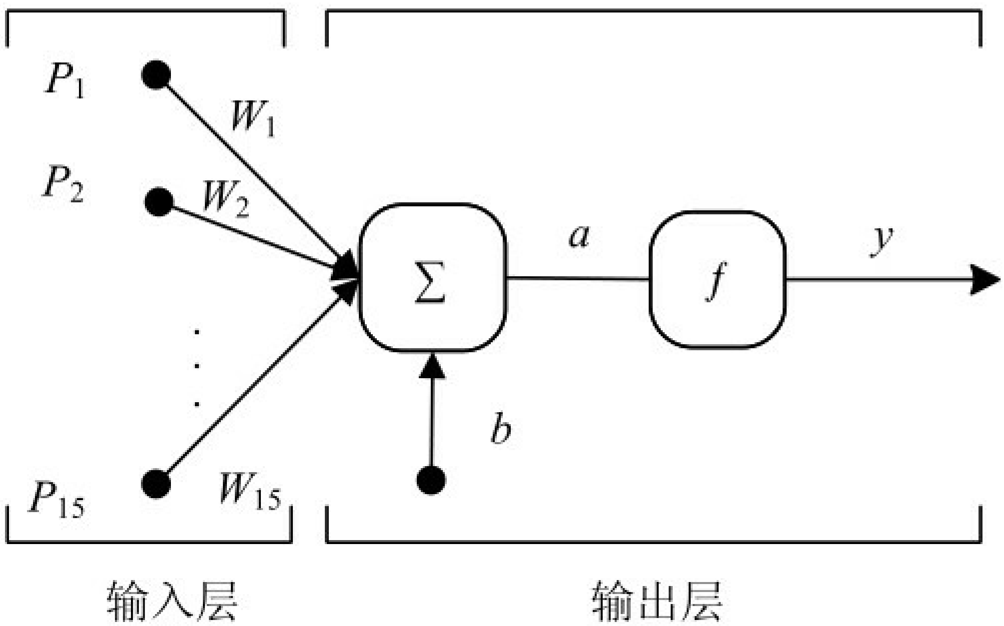


图5-15 单层感知器神经网络结构示意图

创建、训练、存储感知器神经网络的MATLAB程序如下所示。

clear all　　　　%清楚workspace里所有内存变量

clc

pr=[0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1];

%设置输入向量每个元素的最大值和最小值

%%%%%%%%%%%%%%创建感知器神经网络%%%%%%%%%%%%%%%%%%%%%%

net=newp(pr,1);

%%%%%%%%%%%%%%训练感知器神经网络%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%定义15*6的训练样本集输入向量%%%%%%%%%%%%%%%%

P=[1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1]';

T=[1 10 01 0]; %定义目标向量

[net,tr]=train(net,P,T); %训练单层感知器神经网络

iw1=net.iw{1,1}; %输出权值

b1=net.b{1}; %输出阈值

epoch1=tr.epoch; %输出训练过程的每一步长

perf1=tr.perf; %输出每一步训练结果的误差

%%%%%%%%%%%存储训练后的神经网络%%%%%%%%%%%%%%%%%%%%%%%

save net1 net

上述程序运行结果如下所示。

iw1 =

-4 1 -4 4 0 -2 3 -4 2 3 1 2 -1 1 2

b1 =

1

epoch1 =

0 1 2 3 4 5 6 7 8 9

perf1 =

0.5000 0.3333 0.3333 0.3333 0.1667 0.1667 0.3333 0.3333 0.1667 0

感知器训练对话框如图5-16所示。从该图中可以看出，感知器经过9次训练，使输出结果达到目标值。

感知器神经网络训练误差曲线如图5-17所示。

建立的感知器神经网络的仿真程序如下。

clear all

```
clc
load net1net        %加载训练后的神经网络
ptest=[1 1 1 1 0 1 1 0 1 1 0 1 1 1 1;1 1 1 0 1 1 0 1 0 0 1 1 1 0 1]';
%数字0与训练样本一致，数字1与训练样本不一致
a=sim(net, ptest)
```

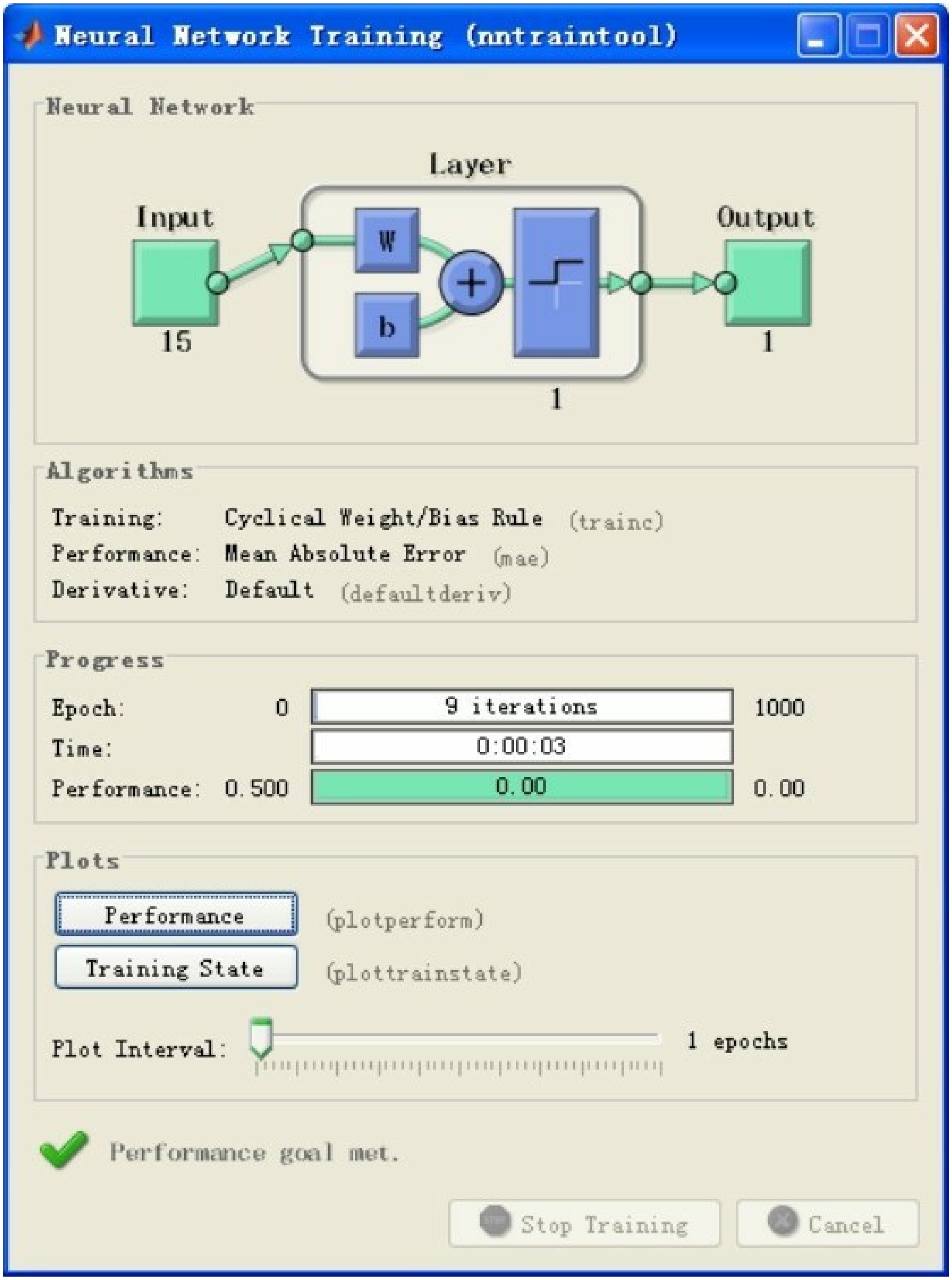


图5-16 感知器训练对话框

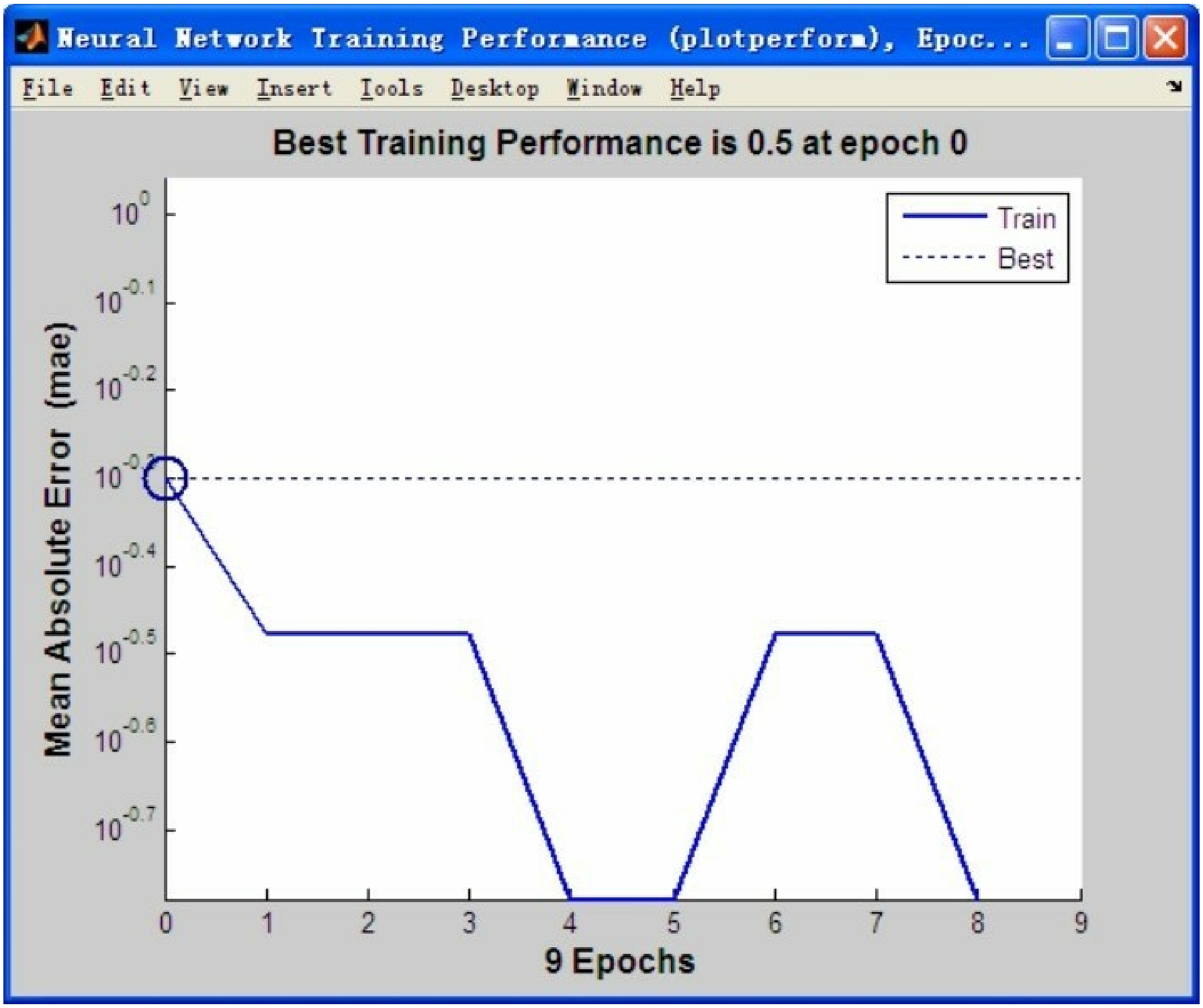


图5-17 感知器神经网络训练误差曲线

运行结果如下。

a =

0

由上面运行结果可以看出，训练后的感知器神经网络，对应正确和不正确的训练样本，都可以得到正确的分类结果，具有一定的容错能力。

【例5-7】给定样本输入向量P、目标向量T及需要进行分类的输入向量组M，设计一个单层感知器，对其进行分类。

解：MATLAB代码如下。

```
clear all
clc
P=[0.5 -1 2;-0.8 1.3 1];
T=[1 1 0];
net=newp([-1 1;-1 1],1);    %建立单层感知器神经网络
h=plotpc(net.iw{1},net.b{1});    %返回画线的句柄
net.trainParam.epochs=50;    %设置训练次数最大为50
net=train(net,P,T);        %训练神经网络
%%%%%%%%%用训练后的感知器网络仿真%%%%%%%%%%%
M=[0.5 1.1 -0.2;-0.2 -0.6 0.6];
Y=sim(net,M);
figure
plotpv(M,Y);            %绘制分界线
h=plotpc(net.iw{1},net.b{1},h)
```

感知器训练对话框如图5-18所示，从图中可以看出，感知器经过两次训练就使输出达到了目标值。

运行程序后，绘制分界线的结果如图5-19所示。由图可以看出，感知器成功地将输入向量进行了分类。

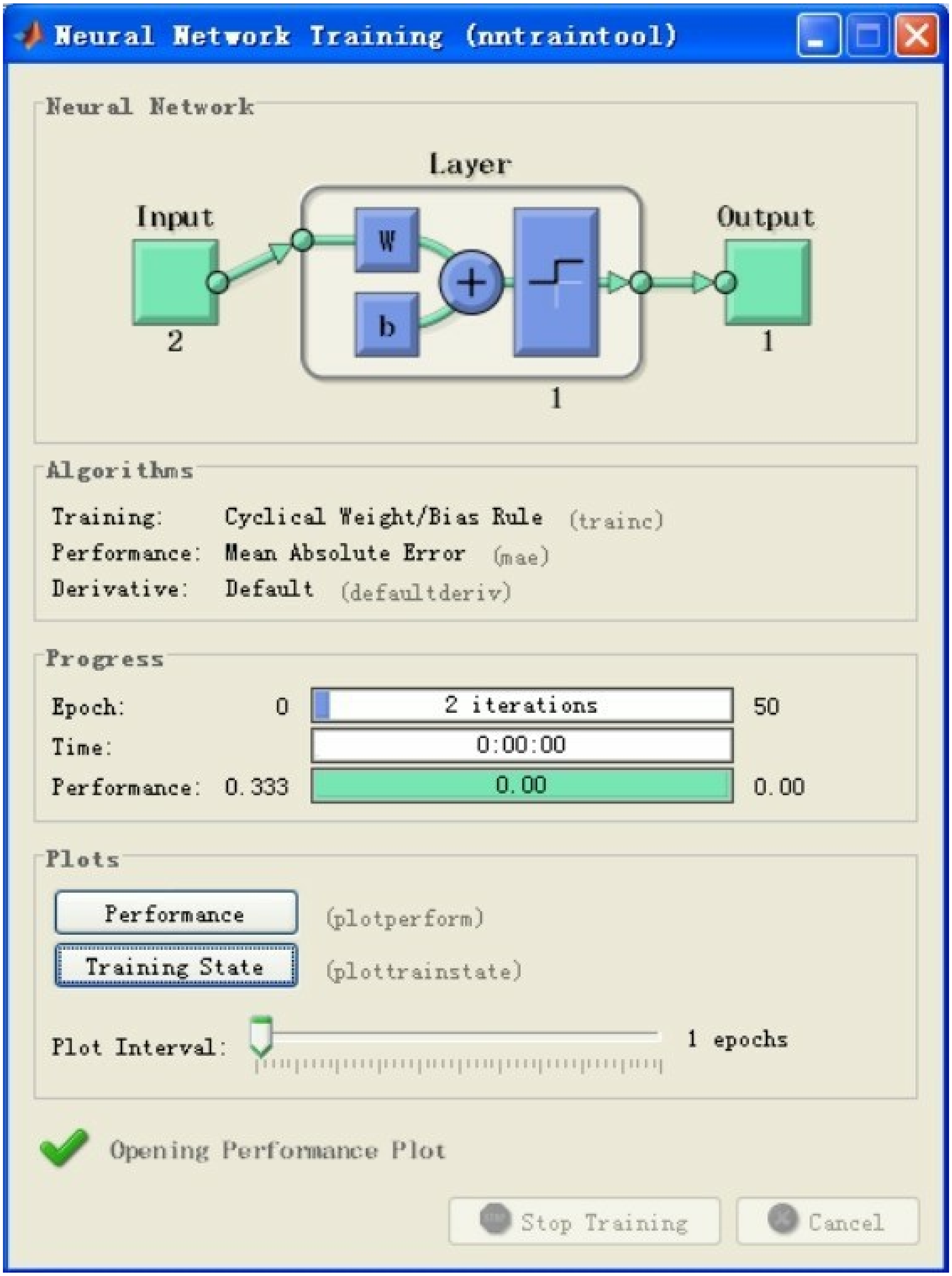


图5-18 感知器训练对话框

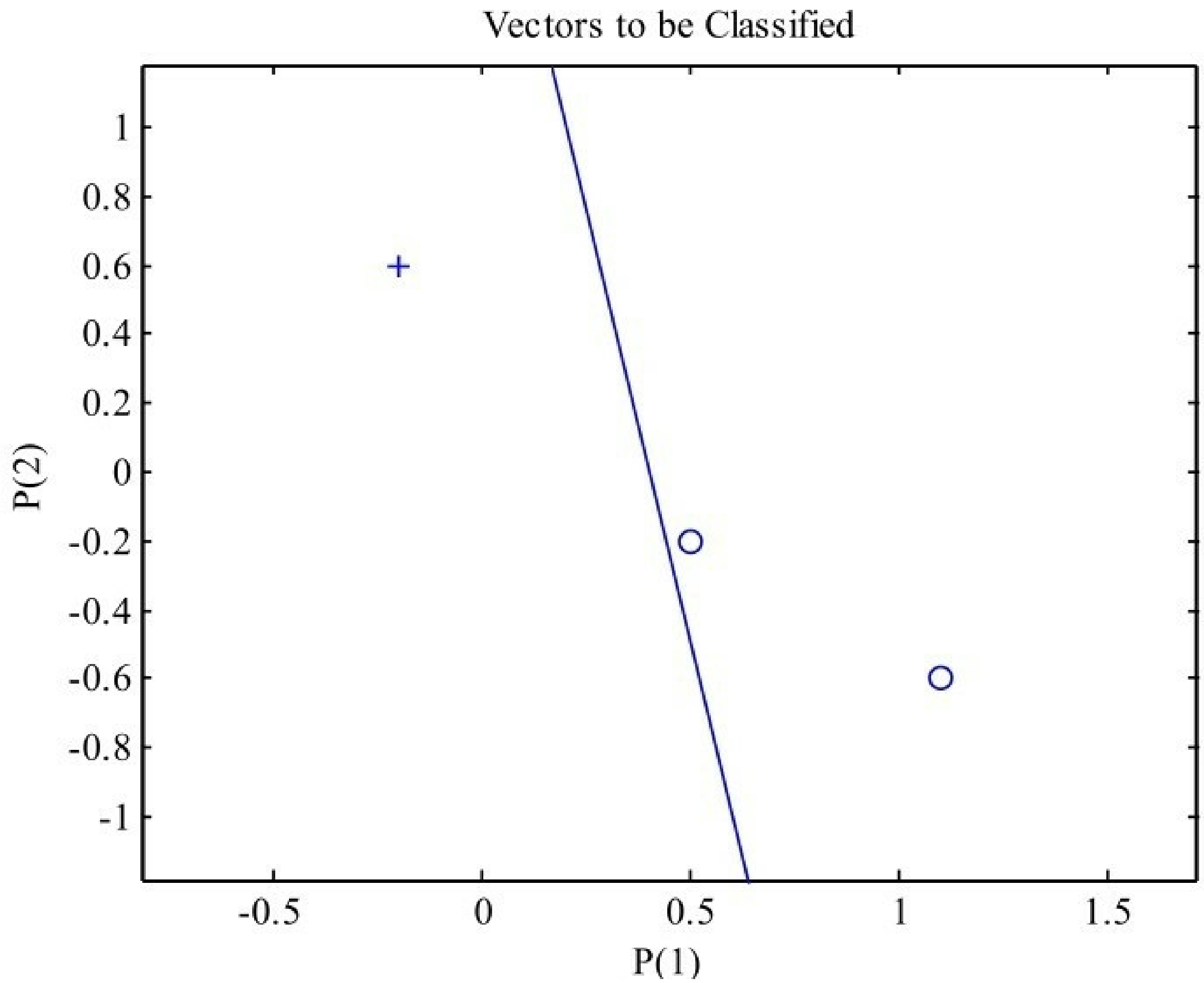


图5-19 程序运行后绘制的分界线

感知器训练结果为。

h =

174.0165

下面讲解一个线性不可分的例子。

假如给定一个双元素的输入向量组P，每列都由6个元素组成，并给定一个目标向量T。利用以下MATLAB代码将其绘制在一个平面上。运行结果如图5-20所示。

```
P=[-0.8 -0.8 1 1.3 2 4;-0.8 0.8 -1 0 3 1]
T=[1 0 0 0 1 1];
plotpv(P,T)
```

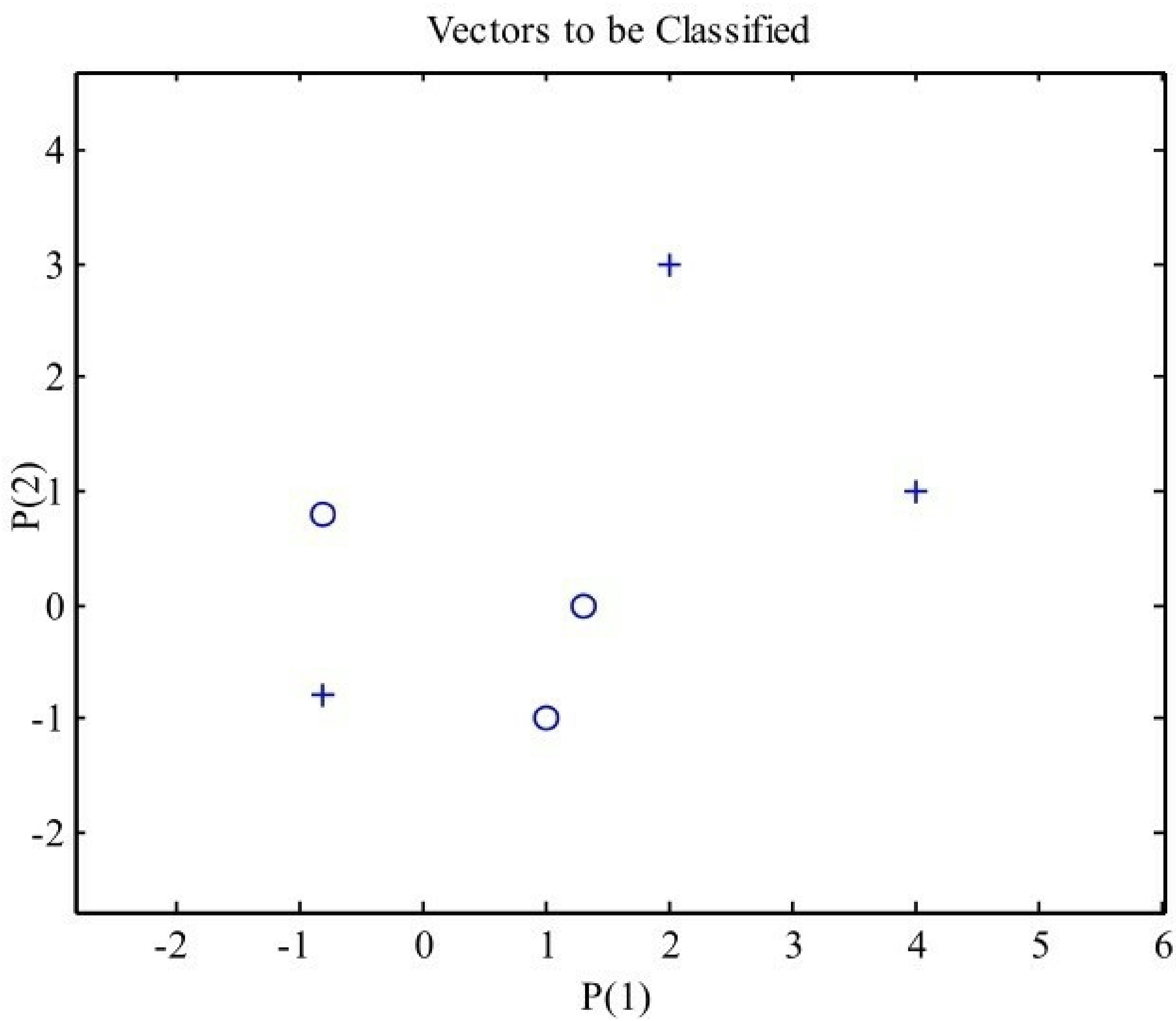


图5-20 样本点分布情况

下面尝试设计一个感知器，该感知器需要对输入向量 P 进行准确的分类。利用函数newp可以创建一个感知器，其程序代码如下所示。

```
net=newp([-35 1;-2 45],1);
```

建立感知器神经网络后，需要对网络进行训练。使用自适应函数adapt 可以对其进行训练。自适应函数设定的循环次数为5次，经过50次迭代后，训练停止。其MATLAB代码如下所示。

```
net.adaptParam.passe=5;
linehandle=plotpc(net.iw{1},net.b{1});
for a=1:50
```

```
   [net,Y,E]=adapt(net,P,T);
linehandle=plotpc(net.iw{1},net.b{1},linehandle);
drawnow;
end
```

对输入样本分类的运行结果如图 5-21 所示。由图 5-21 可知，对于这种线性不可分的模式，利用单层感知器是无法正确分类的。

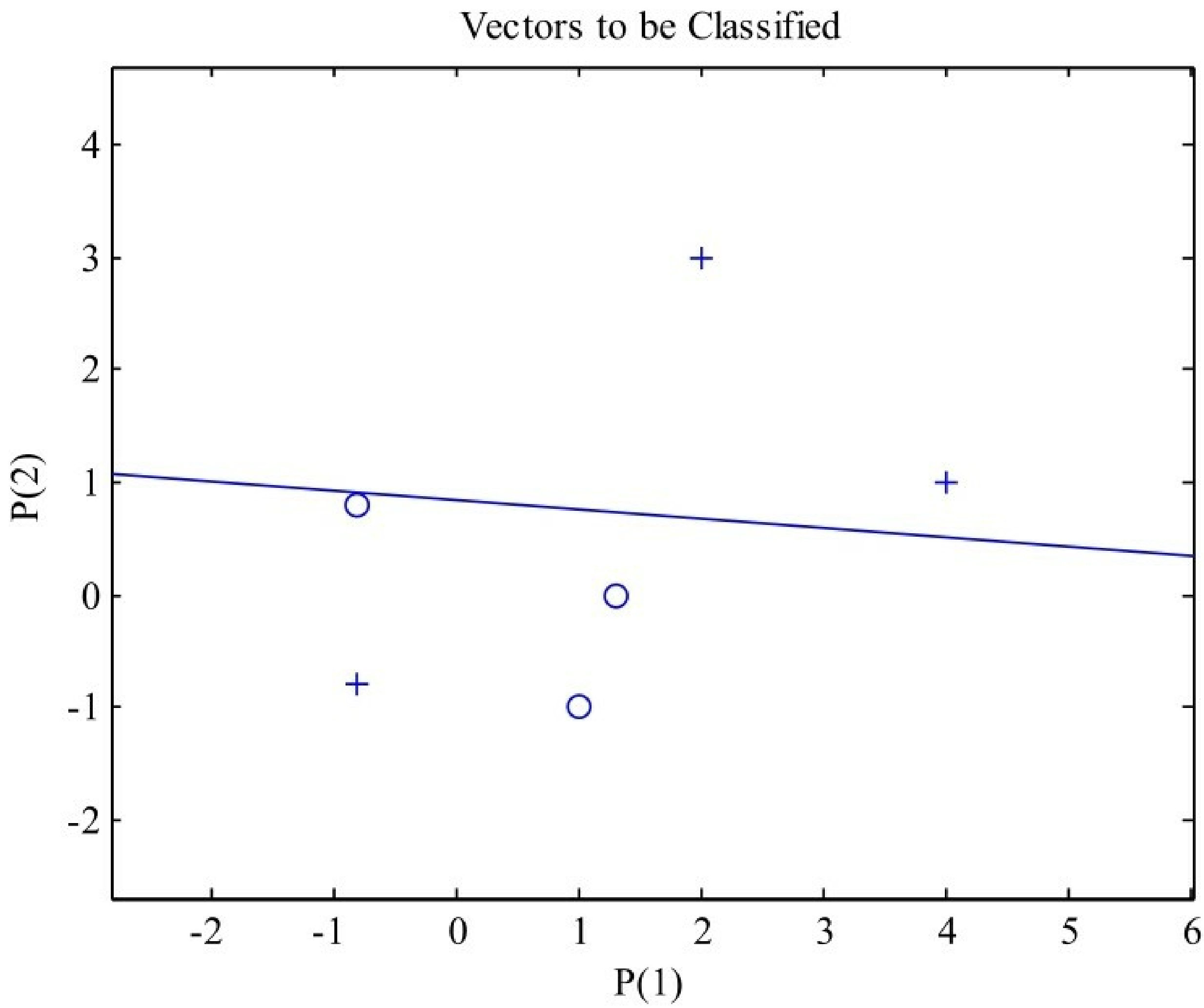


图5-21 对输入样本分类的运行结果

5.4.2 多层感知器MATLAB仿真程序设计

单层感知器由于其结构和学习规则上的局限性，其应用受到了一定

的限制。单层感知器只能对线性可分的向量集合进行分类。为了解决线性不可分的输入向量的分类问题，可以在单层感知器中增加网络层，即多层感知器神经网络。

由于感知器神经网络学习规则的限制，其只能对单层感知器神经网络进行训练。下面介绍一种针对二层感知器神经网络的设计方法。

（1）把感知器的第一层设计作为随机感知器层，且不对其进行训练，而是随机初始化其的权值和阈值。当其接收各输入元素的值时，其输出也是随机的。但其权值和阈值一旦固定下来，对输入向量模式的映射也随之确定了。

（2）以第一层的输入作为第二层感知器的输入，并对应输入向量，确定第二层感知器的目标向量，然后对第二层感知器进行训练。

（3）由于第一层感知器的权值和阈值是随机的，所以其输出也是随机的。在训练过程中，整个网络可能达到训练误差性能指标，也可能达不到训练误差性能指标。当整个网络达不到训练误差性能指标时，需要重新对随机感知器层的权值和阈值进行初始化赋值，可以将其初始化函数设置为随机函数，然后用init函数重新初始化。

（4）如果程序第一次运行结果达不到设计要求，那么需要重复训练，直到运行结果达到要求为止。

【例5-8】用二层感知器神经网络来实现异或函数。

解：异或真值表见表5-4。

表5-4 异或真值表

输入 p_1	输入 p_2	输出 y
0	0	0
1	0	1
0	1	1
1	1	0

若把异或问题看成 p_1-p_2平面上的点，则点 $Y_0(0,0)$、$Y_1(1,1)$表示输出为 0 的两个点， $Z_0(1,0)$、$Z_1(0,1)$表示输出为1的两个点，如图5-22所

示。

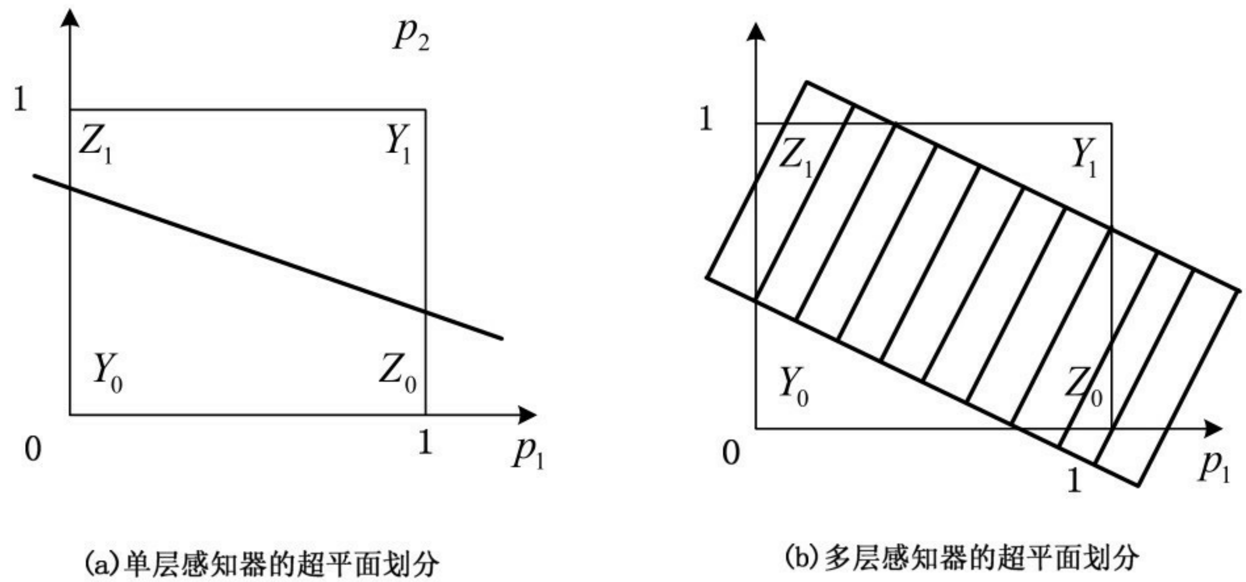


图5-22 异或问题的图形表示

从图5-22中可以看出，无论在平面上怎么用一条直线，也不能将输出为0和输出为1的两种模式分开，而用两条直线就能将输入为0和1的两种模式分开。

根据以上分析，如果用两层感知器，每层感知器可以构成一条直线来划分，这样就可以解决模拟异或函数的问题。

图5-23所示为两层感知器神经网络结构图，其中第一层神经元数目设计为4，其权值和阈值是随机的，它的输出作为第二层感知器的输入。第二层感知器神经元个数为1。

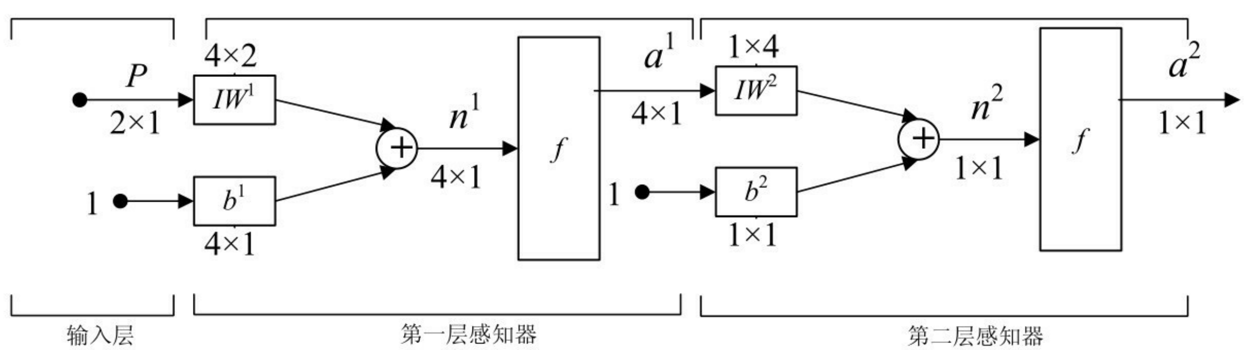


图5-23 两层感知器神经网络结构图

二层感知器神经网络的MATLAB实现代码如下所示。

```
clear all
clc
%%%%%%%%%%%第一层感知器初始化%%%%%%%%%%%%%%%%%%%%%
PR1=[0 1;0 1];
net1=newp(PR1,4);
net1.inputweights{1}.initFcn='rands';
net1.biases{1}.initFcn='rands';
net1=init(net1);
IW1=net1.iw{1}
B1=net1.b{1}
%%%%%%%%%%%第一层感知器仿真%%%%%%%%%%%%%%%%%%%%%%
P1=[0 0;0 1;1 0;1 1]';
[A1,Pf]=sim(net1,P1);
%%%%%%%%%%%第二层感知器初始化%%%%%%%%%%%%%%%%%%%%%%
PR2=[0 1;0 1;0 1;0 1];
net2=newp(PR2,1);
%%%%%%%%%%%训练第二层感知器%%%%%%%%%%%%%%%%%%%%%%%%
net2.trainParam.epochs=500;
net2.trainParam.show=5;
P2=ones(4,4);
P2=P2.*A1;
T2=[0 1 1 0];
```

```
[net2,TR2]=train(net2,P2,T2);
epoch2=TR2.epoch
perf2=TR2.perf
IW2=net2.iw{1}
B2=net2.b{1}
%%%%%%%%%%存储训练后的网络%%%%%%%%%%%%%%%%%%%%%%%
save net3 net1 net2
```

因为第一层感知器的权值和阈值是随机的，所以其输出也是随机的，整个网络可能达到训练误差的指标，也可能达不到训练误差指标。当第一层感知器达不到误差指标时，需要重新对其权值和阈值进行初始化来复制。

在本题程序中，通过将其初始化函数设置为随机函数，然后用 init 函数重新初始化来实现。该程序第一次运行的结果可能达不到设计要求，需要反复运行，直至达到要求为止。这也导致了每次训练得到的结果不一定相同。

这样做的原因是设计受到感知器的学习算法的限制。对于多层网络，本书将在后面章节加以介绍，这里不做论述。

在多次运行题中程序后，选择一种训练达到指标的运行结果如下所示。

```
IW1 =
   -0.6756   -0.6687
   0.5886    0.2040
   -0.3776   -0.4741
   0.0571    0.3082
B1 =
   0.1376
```

    -0.0612

    -0.9762

    -0.3258

epoch2 =

    0   1   2   3

perf2 =

    0.5000   0.5000   0.5000   0

IW2 =

    -1   1   0   -2

B2 =

    0

感知器训练对话框如图5-24所示，训练得到的误差曲线如图5-25所示。

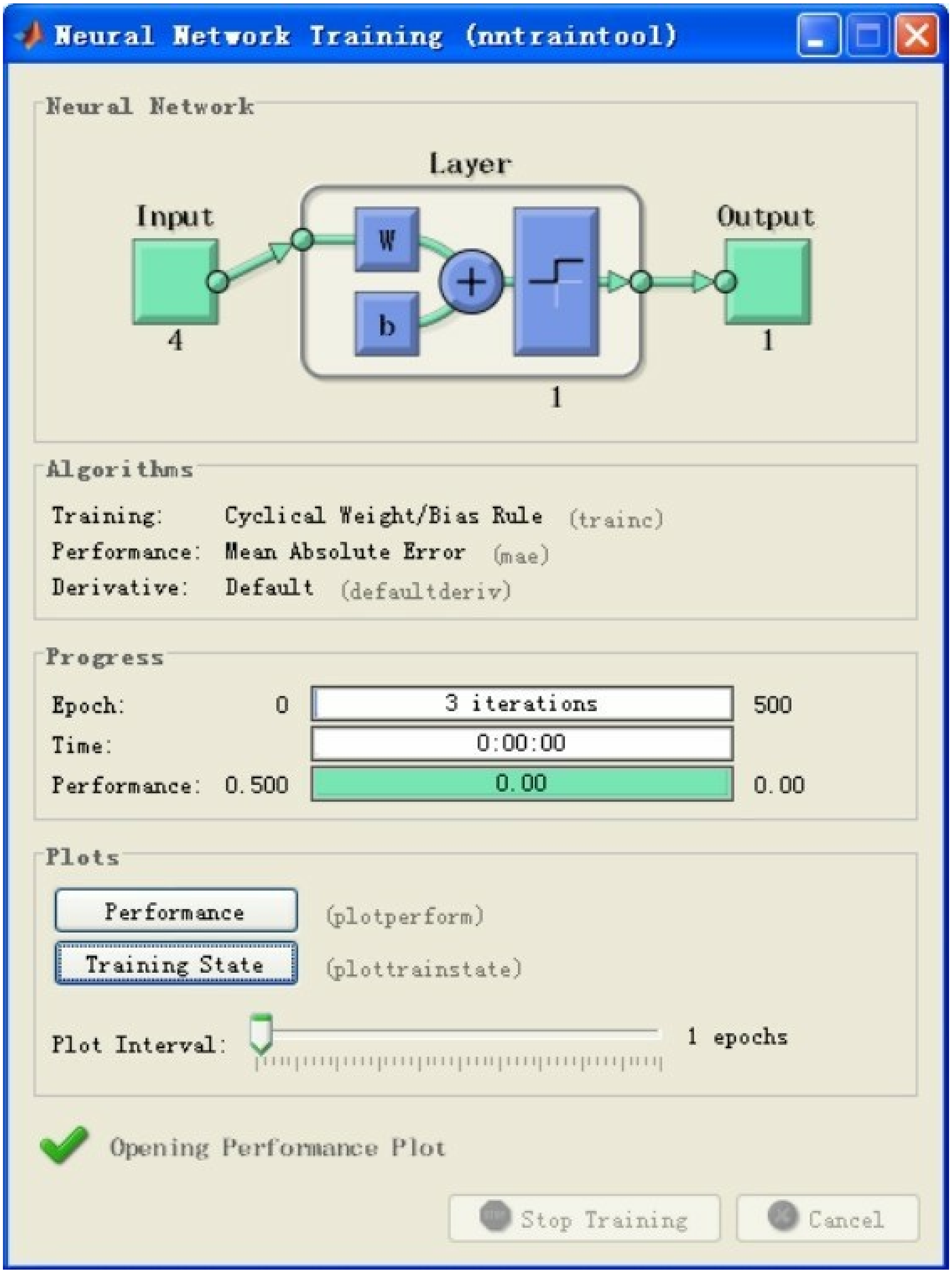


图5-24 感知器训练对话框

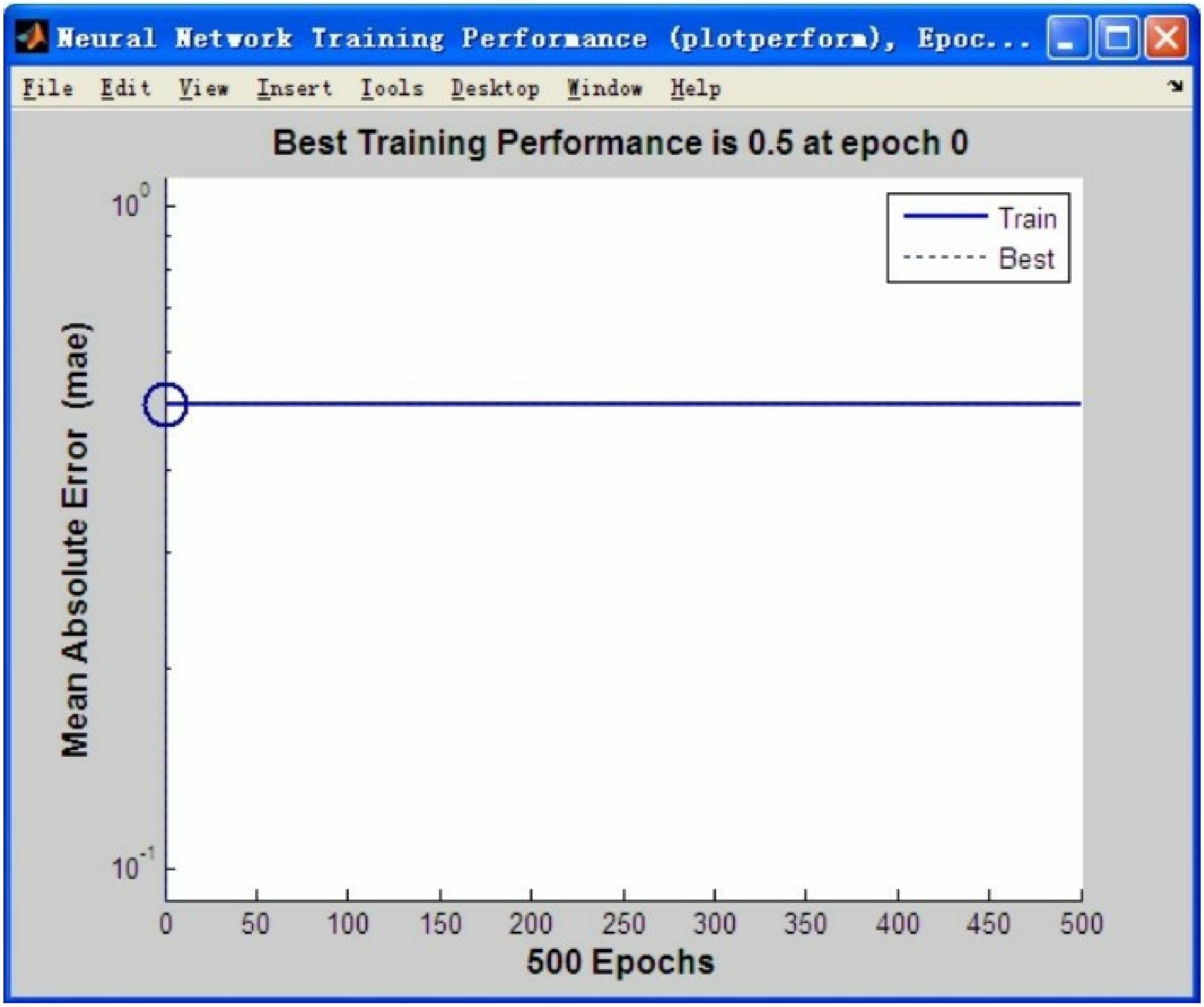


图5-25 训练得到的误差曲线

两层感知器神经网络仿真的MATLAB程序如下所示。

```
load net3 net2 %加载训练后的感知器神经网络
P1=[0 0;0 1;1 0;1 1]';
A1=sim(net1,P1);%第一层感知器仿真
P2=ones(4,4);
P2=P2.*A1;
A2=sim(net2,P2) %第二层感知器仿真
```

输出仿真结果如下。

A2 =

0 1 1 0

可以看出，所涉及的网络可以正确模拟“异或”函数的功能。

5.5 本章小结

感知器是最早被提出的一种神经网络模型。本章详细介绍了感知器的模型、初始化、学习规则等原理，还对感知器的局限性和其工具箱函数进行了简单说明。在本章的最后部分，根据例题详细说明了感知器MATLAB仿真程序设计方法。

第6章 线性神经网络

线性神经网络由一个或多个线性神经元构成，它是最简单的一种神经元网络。1960年斯坦福大学教授B.Widrow和M.Hoff提出的自适应线性神经元是线性神经网络最早的典型代表，其学习算法是least mean squares（简称LMS）算法。

线性神经网络主要用于函数逼近、信号处理滤波、预测、模式识别等方面，本章主要介绍线性神经网络原理、工具箱函数及MATLAB仿真。

学习目标：

·熟悉线性神经网络原理

·掌握线性神经网络工具箱函数的使用

·掌握线性神经网络MATLAB 仿真程序设计

6.1 线性神经网络原理

线性神经网络与感知器神经网络的不同之处在于，每个神经元的传递函数为线性函数，因此感知器神经网络的输出只能是0或1，线性神经网络的输出可以取任意值。

6.1.1 线性神经网络模型

在介绍线性神经网络模型之前，先介绍线性神经元模型，其模型如图6-1所示。

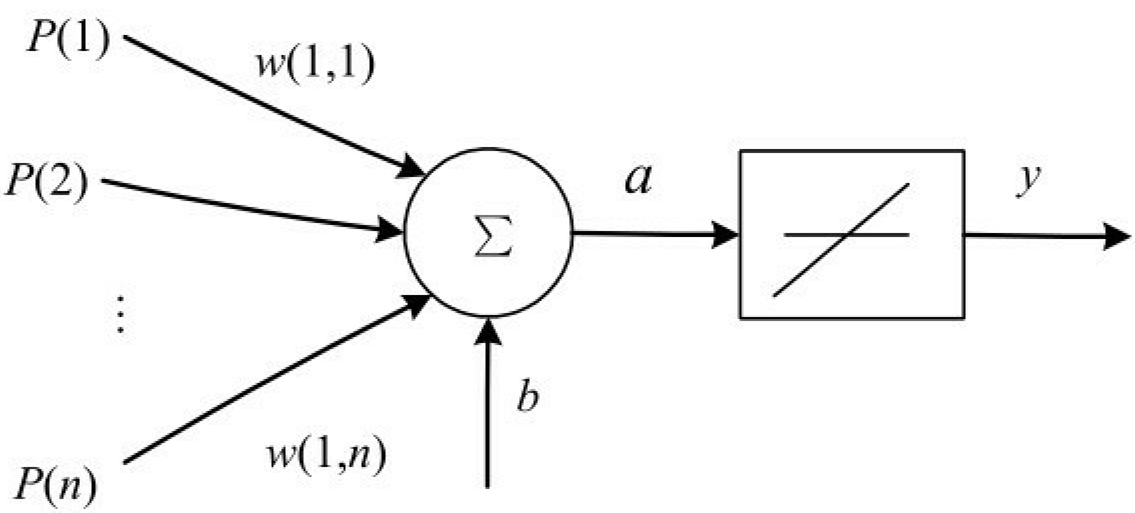


图6-1 线性神经元模型

线性神经元的输入为P，权值和阈值分别为w和b，线性神经元模型的输出为y，其传递函数为线性传递函数purelin，如图6-2所示。

由于线性神经网络中神经元的传递函数为线性函数，网络输入和输出之间存在简单的比例关系，因此，对应单个线性神经元，可以根据下面公式计算。

$$y=purelin(w\cdot p+b)$$

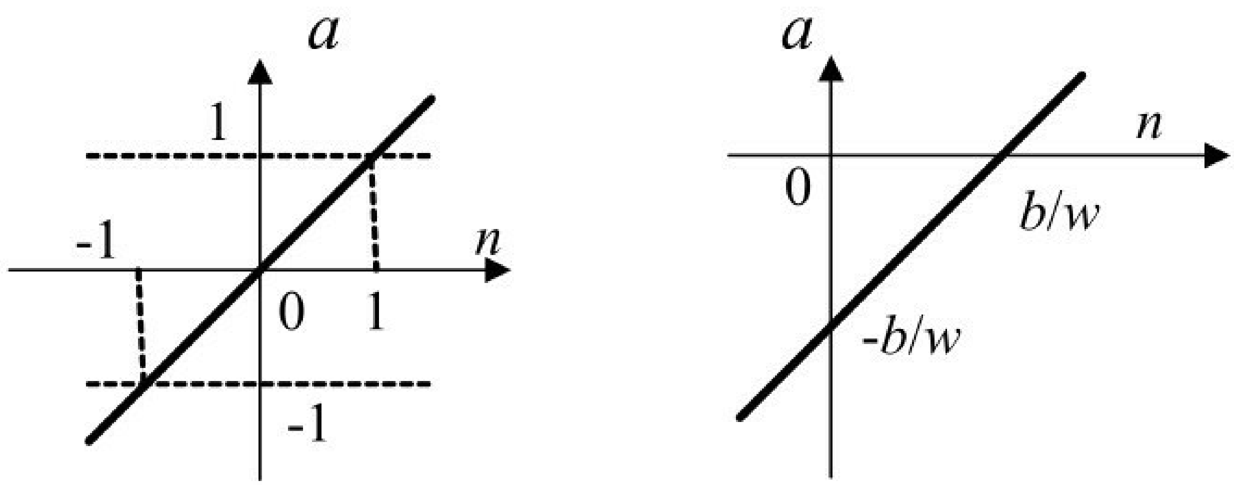


图6-2 线性传递函数purelin

在MATLAB中，可用如下所示代码实现生产线性神经网络和计算网络的输出。

```
net=newlind(P,T)
y=sim(net,P)
```

线性神经网络模型是具有多个神经元的模型。具有n个输入的单层（m个神经元）线性神经元网络的两种形式如图6-3所示。图中权值矩阵为w，阈值向量为b，这种网络也称为Madaline网络。

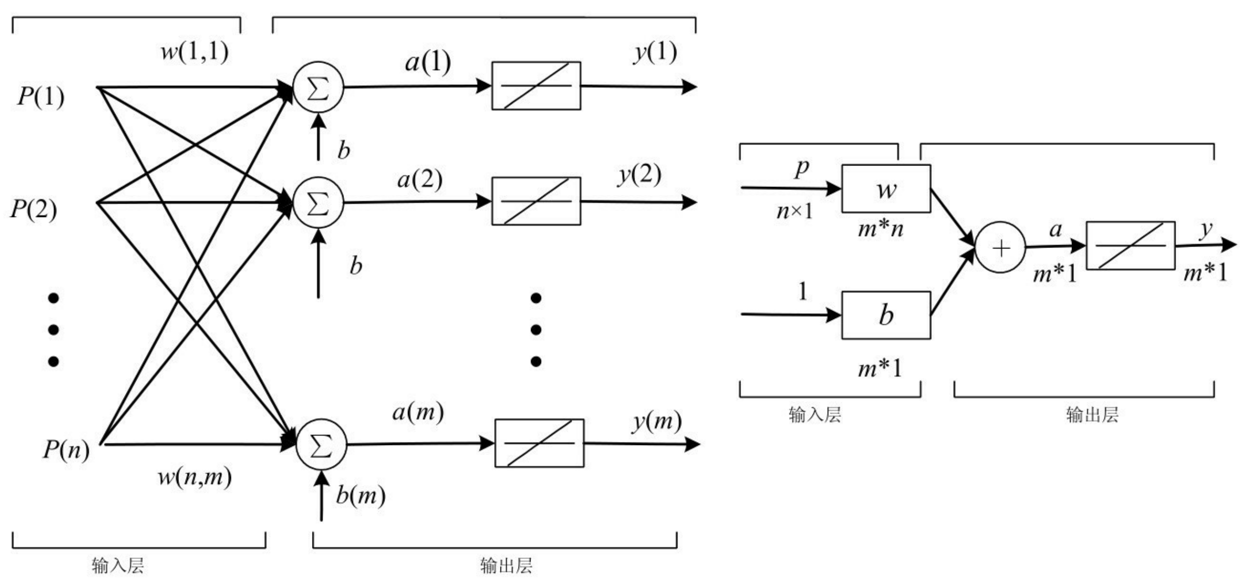


图6-3 线性神经网络的两种方式

Widrow-Hoff 学习规则只能训练单层的线性神经元网络，但这一点并不影响单层线性神经网络的应用，因为对每一个多层线性神经网络而言，都可以设计出一个性能相当的单层线性神经网络。

6.1.2 线性神经网络初始化

与感知器神经网络相同，线性网络的初始化也采用MATLAB神经网络工具箱中的init()函数。其用法如下所示。

```
net=init(net);
```

使用上述语句，网络的权值和阈值被设置为0。MATLAB神经网络工具箱中函数newlin和newlind均可产生一个单层线性网络，在这两个函数中使用了init()函数，因此得到的线性网络已经是初始化的。

【例6-1】用newlin函数设计一个单输入的线性神经网络，要求学习速率为0.05。网络设计好之后，要求显示其权值和阈值。

解：设计一个线性神经网络，其输入范围设定为[0,1]。MATLAB代码如下所示。

```
net=newlin([0 1],1,[ ],0.05);
net.iw{1}
net.b{1}
```

执行以上语句后可以得到一个初始化后的线性网络，其权值和阈值均为 0。对上述网络在给定的输入和目标下进行训练，得到相应的权值和阈值。如果想重新对网络进行初始化，输入以下代码。

```
net=init(net)
net.iw{1}
net.b{1}
```

这时，网络的权值和阈值又重新被初始化为0。

6.1.3 线性神经网络学习规则

对于线性神经网络，可以不经过训练而直接计算网络的权值和阈值。如果网络有多个零误差解，则取最小的一组权值和阈值；如果网络不存在零误差解，则取网络的误差平方和最小的一组权值和阈值。

因为线性系统有唯一的误差最小值，所以可通过给定的输入向量和目标向量，计算出实际输出向量和目标向量的误差最小值。

当不能直接求出网络权值和阈值时，线性神经网络可以采用使均方误差最小的学习规则，即LMS算法。该规则是一种沿误差的最陡下降方向对前一步权值进行修正的方法。

对于n个训练样本有

$$\{p_1,t_1\},\{p_2,t_2\},\cdots,\{p_n,t_n\}$$

LMS学习规则的基本思想是要寻找最佳的权值和阈值，使得各个神经元的输出均方误差最小。

神经元的均方误差计算公式有

$$mse=\frac{\sum_{i=1}^{n}(t_i-y_i)^2}{n}=\frac{\sum_{i=1}^{n}e_i^2}{n}$$

其中，n为训练样本数，y为神经元输出值，t为神经元输出的目标值。

为了找到使每个神经元输出均方误差最小的权值和阈值，以x代表权值或者阈值，对x求偏导

$$\frac{\partial mse}{\partial x} = \frac{\partial(\sum_{i=1}^{n} e_i^2)}{\partial(n)}$$

令上式等于 0，则可以求出 mse 的极值点。极值点可以是极大值，也可以是极小值。因为mse只能是正值，所以极值点肯定是极小值。但是，按照上式求偏导，计算很繁杂，特别是当输入向量的维数很高时，计算更加麻烦。一般采用搜索优化法：假设获得第 i 次训练得到权值或阈值x(i)，然后找出该点在mse-x曲面上的最陡下降方向，再沿此方向对权值进行修正。

对于单个线性神经元$a(i) = W_{1,j}(i)p + b(i)$，有下面等式

$$\frac{\partial e(i)}{\partial W_{1,j}(i)} = \frac{\partial[t - a(i)]}{\partial W_{1,j}(i)} = \frac{\partial[t - [W_{1,j}(i)p + b(i)]]}{\partial W_{1,j}(i)} = -p$$

同理，

$$\frac{\partial e(i)}{\partial b(i)} = \frac{\partial[t - a(i)]}{\partial b(i)} = \frac{\partial[t - [W_{1,j}(i)p + b(i)]]}{\partial b(i)} = -1$$

为避免求均方误差梯度的麻烦，以误差平方的梯度代替均方误差的梯度，则

$$\frac{\partial mse(i)}{\partial W_{1,j}(i)} = \frac{\partial e(i)^2}{\partial W_{1,j}(i)} = 2e(i)\frac{\partial e(i)}{\partial W_{1,j}(i)} = -2e(i)p$$

$$\frac{\partial mse(i)}{\partial b(i)}=\frac{\partial e(i)^2}{\partial b(i)}=2e(i)\frac{\partial e(i)}{\partial b(i)}=-2e(i)$$

所谓最陡梯度下降就是梯度的反方向，即

$$W_{1,j}(i+1)=W_{1,j}(i)+2\alpha e(i)p$$
$$b(i+1)=b(k)+2\alpha e(i)$$

其中，α是神经网络的学习率，即决定权值和阈值的收敛速度和稳定性的参数。学习率越大，学习的速度就越快。

注意：过大的学习率会使神经网络的修正过度，使其不稳定，反而使得误差更大。所以在建立神经网络时，要选择合适的学习率。

上面讨论的是单个线性神经元的LMS算法，如果将其推广到多个线性神经元的情况，可以得到公式

$$W(i+1)=W(i)+2\alpha e(i)p^T$$
$$b(i+1)=b(k)+2\alpha e(i)$$

线性神经网络的LMS学习规则表现为曲面上的梯度下降，是建立在均方误差最小化基础上的。由于误差与权值构成的抛物面只有一个极小值点，因此LMS算法可以保证误差函数最小，但是这需要无限次的学习。

在实际应用中，有限次的学习结果只能得到近似解，只有当输入模式与线性无关时， LMS算法才可以使误差函数为零。

提示：在MATLAB工具箱中，采用LMS算法的学习函数为learnwh。

还有一种W—H学习规则，它是由威德罗和霍夫提出的用来修正权矢量的学习规则，所以用他们两人姓氏的第一个字母来命名。

W—H 学习规则可以用来训练一定网络的权值和偏差，使之线性地逼近一个函数式而进行模式联想（Pattern Association）。

定义一个线性网络的输出误差函数为

$$E(W,B)=\frac{1}{2}[T-A]^2=\frac{1}{2}[T-WP]^2$$

由上式可以看出：线性网络具有抛物线型误差函数所形成的误差表面，所以只有一个误差最小值。通过W—H学习规则来计算权值和偏差的变化，并使网络误差的平方和最小化，总能够训练一个网络的误差趋于这个最小值。

E(W，B)只取决于网络的权值及目标矢量。我们的目的是通过调节权矢量，使 E(W， B)达到最小值。所以在给定E(W，B)后，利用W—H学习规则修正权矢量和偏差矢量，使E(W，B)从误差空间的某一点开始，沿着E(W，B)的斜面向下滑行。

根据梯度下降法，权矢量的修正值正比于当前位置上 E(W，B)的梯度，对于第 i 个输出节点有

$$\Delta w_{ij}=-\eta\frac{\partial E}{\partial w_{ij}}=\eta[t_i-a_i]p_j$$

或表示为

$$\Delta w_{ij}=\eta\delta_i p_j$$

$$\Delta b_i=\eta\delta_i$$

这里 δ_i定义为第i个输出节点的误差

$$\delta_i=t_i-a_i$$

上式称为 W—H 学习规则，又叫δ规则，或为最小均方差算法

（LMS）。W—H 学习规则的权值变化量正比于网络的输出误差及网络的输入矢量。它不需求导数，所以算法简单，又具有收敛速度快和精度高的优点。

η为学习速率。在一般的实际运用中，实践表明，η通常取一接近1的数，或取值为

$$\eta = 0.99 \times \frac{1}{\max[\det(p \times p^T)]}$$

这样的选择可以达到既快速又正确的结果。

学习速率的这一取法在神经网络工具箱中用函数maxlinlr来实现。上式可实现为

$$lr = 0.99 \times \mathrm{maxlinlr}(P,1)$$

其中lr为学习速率。

W—H 学习规则的函数以 learnwh.m 来实现，另外，加上线性自适应网络输出函数purelin.m，可以写出W—H学习规则的计算公式为

A＝purelin(W·P)；

E＝T−A；

[dW，dB]＝learnwh(P,E,h)；

W＝W+dW；

B＝B+dB；

采用W—H规则训练自适应线性元件使其能够得以收敛的必要条件是被训练的输入矢量必须是线性独立的，且应适当地选择学习速率以防止产生振荡现象。

6.1.4 线性神经网络的训练

用train函数可以完成对线性神经网络的训练。这个训练过程，实际

上是根据所给出的“输入—目标”样本矢量集，调用神经网络生产时的权值和阈值。因为学习函数learnwh对神经网络不断进行调节，最终使网络的输出接近目标输出的过程。

下面代码给出了一个典型的具有三维输入线性神经元的训练过程。

```
clear all;
clc
P=[1 2 1;2 4 3;-1 0 5];
T=[0 1 1];
net=newlin([1 2;2 4;-1 5],1);
net.trainParam.goal=0.01;
[net,tr]=train(net,P,T);
net.iw{1,1}
net.b{1}
```

运行程序后显示结果如下，网络训练误差曲线如图6-4所示。

```
ans =
   0.1899    0.2320    0.1090
ans =
   -0.4031
```

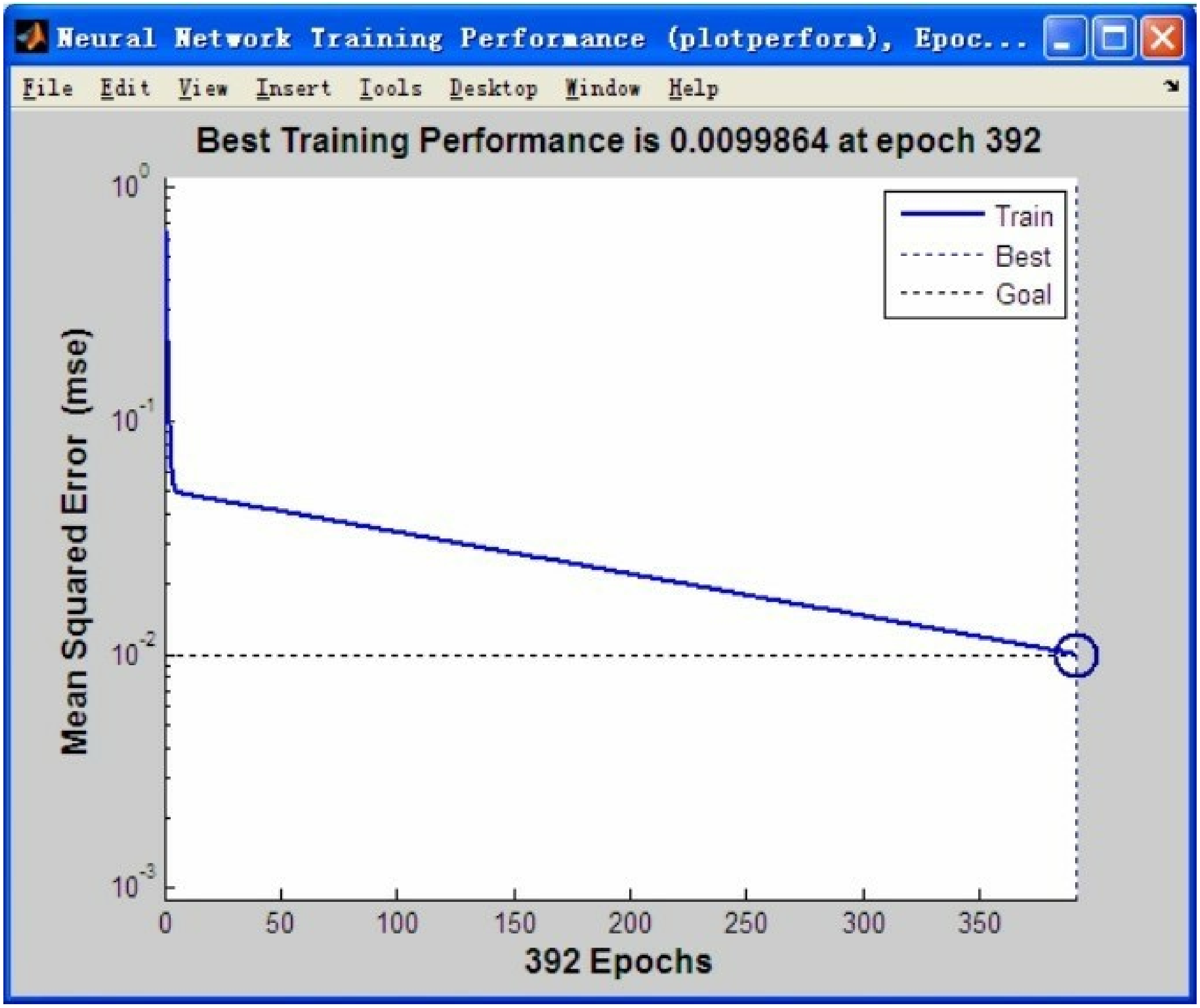


图6-4 网络训练误差曲线

利用sim等函数对训练好的线性神经网络进行仿真和误差分析。

```
>> a=sim(net,P)
a =
   0.1418    0.9048    1.0280
>> error=T-a
errot =
   -0.1418    0.0952    -0.0280
>> m=mse(error)
m =
```

0.0100

【例6-2】设计自适应线性网络，实现从输入矢量到输出矢量的变换关系。其输入矢量和输出矢量分别如下。

P＝[1.0 -1.2]

T＝[0.5 1.0]

解：用自适应线性网络求解问题时，设计者要确定期望误差值，以及最大循环次数。自适应线性网络MATLAB代码如下所示。

```
P＝[1 -1.2]；
T＝[0.5 1]；
[R，Q]＝size(P)；
[S，Q]＝size(T)；
[W，B]＝rands(S，R)；
max_epoch＝20；          %最大循环次数
err_goal＝0.001；        %期望误差
1r＝0.4*maxlinlr(P)；          %最佳学习速率
disp_freq＝1；          %设置显示频率
TP＝[disp_freq max_epoch    err_goal    lr]；       %设置参数变量TP
[W，B，epochs，error]＝trainwh(W，B，P，T，TP)  %进行线性网络权值训练
```

在随机初始值为：W_0＝−0.9309；B_0＝−0.8931的情况下，经过12次循环训练后，网络的输出误差平方和达到0.000949，网络的最终权值如下。

W＝-0.2354；

B＝0.7066

实际上，对于上述例题来说，它存在一个精确解，且可以用解二元一次方程的方式将P 和T 值分别对应地代入方程T＝W·P+B 得

$$\begin{cases} W + B = 0.5 \\ -1.2W + B = 1 \end{cases}$$

可解出e＝T−A＝0的解如下。

W＝-0.2273；

B＝0.7273

由此看出，对于特别简单的问题，采用自适应线性网络的训练不一定能够得到足够精确的解。因为当训练误差达到期望误差值后，训练即被终止。

对于具有零误差的自适应线性网络，即输入/输出矢量对存在着严格的线性关系，此时的自适应线性网络的设计可以采用工具箱中另外一个名为solvelin的函数。

在上例中，可简单使用下面的命令。

[W，B]＝solvelin(P，T)

立即得到精确解。

W＝−0.2273

B＝0.7273

然后可用simulin函数来检测所设计的网络。

```
>>A＝simulin(P，W，B)
A＝
   0.5000 1.0000
```

还可以用sumsqr函数来求出误差平方和。

```
>>SSE＝sumsqr(T-A)
SSE＝
0
```

可见所设计网络的误差为零。

6.2 线性神经网络工具箱函数

MATLAB神经网络工具箱为线性网络提供了大量用于网络设计、创建、分析、训练和仿真等的函数。下面对这些函数的功能和使用方法进行详细介绍。

在MATLAB中与线性神经网络相关的工具箱函数见表6-1。

表6-1 与线性神经网络相关的工具箱函数

函数名称	功　　能
newlind	设计一个线性网络
newlin	构造一个线性网络
purelin	线性传递函数
dotprod	极值点积函数
netsum	网络输入求和函数
initlay	网络某一层的初始化函数
initwb	网络某一层的权值和阈值的初始化函数
initzero	零权值和阈值初始化函数
init	网络的初始化函数
mae	求平均绝对误差性能函数
learnwh	LMS 学习规则
adaptwb	网络权值和阈值的自适应函数
trainwb	网络的权值和阈值训练函数
train	神经网络训练函数
maxlinlr	线性层的最大学习率
errsurf	计算误差性能曲面
sim	网络仿真函数

对于表6-1中所述函数，下面选取其中比较重要的几个函数进行详细介绍。

1.newlind

该函数的作用是设计一个线性网络，其使用格式如下。

net=newlind(P,T)

其中，P为输入向量。

T为目标向量。

该函数返回一个将输入P设计为输入T的具有最小均方误差和的线性网络。

通过求解下面的线性方程，newlind函数可以根据输入P和目标T计算一个线性网络的权值和阈值

$$[w,b]\cdot[P;\mathrm{ones}]=T$$

【例6-3】设计一个线性网络，具有如下给定的输入P和输出目标T。

P=[0 1 3 5];

T=[3 -1 2 6];

解：使用newlind函数来设计符合题目要求的线性网络。MATLAB代码如下。

```
clear all
clc
P=[0 1 3 5];
T=[3 -1 2 6];
net=newlind(P,T);
Y=sim(net,P)
```

该程序运行结果如下。

```
Y =
   0.5932    1.4407    3.1356    4.8305
```

2.newlin

该函数的作用是建立一个线性网络。其使用格式如下所示。

net=newlin(PR,S,ID,LR)   %建立一个新的线性网络

net=newlin（PR，S，0，P）   %返回一个对输入P具有最大稳定学习率的线性网络

其中，PR为输入元素的最大、最小值矩阵。

S为输出向量的个数。

ID为输入延迟向量，默认值为[0]。

LR为网络学习率，默认值为0.01。

P为输入向量的矩阵。

【例6-4】设计一个单输入（输入范围为[-2　2]）单个神经元的线性网络，输入延迟为0和1，学习率为0.05。输入向量P和目标向量T分别如

下。

```
P1=[0 -2 1 2 0 -1 1 2 0 1];
T1=[1 -1 0 2 1 0 0 1 2 1]
```

根据以上输入和目标向量对网络进行训练和仿真。

解：MATLAB代码如下。

```
net=newlin([-2 2],1,[0 1],0.05);
P1=[0 -2 1 2 0 -1 1 2 0 1];
Y=sim(net,P1)
```

仿真结果如下。

```
Y =
   0    0    0    0    0    0    0    0    0
```

根据目标向量T，对网络进行自适应。

```
T1=[1 -1 0 2 1 0 0 1 2 1];
[net,Y,E,Pf]=adapt(net,P1,T1);
Y
```

运行程序后，得到仿真结果如下所示。

```
Y =
   0.2500    -0.6500    0.7000    1.1500    0.2500    -0.2000    0.7000
1.1500    0.2500    0.7000
```

【例6-5】使用例6-4中得到的Pf作为初始条件，使建立好的线性网络适应一个新的输入向量P2=[2 0 1 -2 0 -2 0 1 1 -1]和目标向量T2=[1 -1 1 -2 1 -1 1 1 2 0]。

解：根据新的输入向量和目标向量，对线性网络进行训练，MATLAB代码如下。

```
P2=[2 0 1 -2 0 -2 0 1 1 -1];
T2=[1 -1 1 -2 1 -1 1 1 2 0];
[net,Y,E,Pf]=adapt(net,P2,T2);
```

Y

上述代码运行结果如下。

Y =

1.2650   0.2850   0.7750   -0.6950

0.77500.2850   -0.6950   0.2850   0.7750 -0.2050

假如需要对上面的线性网络进行重新初始化，以便得到新的权值和阈值。用例 6-4 和例 6-5 中的输入向量训练这个初始化的线性网络，使得其输出误差达到 0.58，最大训练次数设置为50。可以编写以下代码。

```
net=init(net);
P3=[P1 P2];
T3=[T1 T2];
net.trainParam.epochs=50;
net.trainParam.goal=0.5;
net=train(net,P3,T3);
Y=sim(net,[P1,P2])
```

得到的仿真结果如下。

Y =

Columns 1 through 10

0.36200.3620   -0.7800   0.9330   1.5040   0.3620   -0.2090

0.9330   1.5040 0.9330

Columns 11 through 20

1.5040   0.3620   0.9330   -0.7800   0.3620   -0.7800   0.3620

0.9330   0.9330 -0.2090

网络训练误差曲线如图6-5所示。

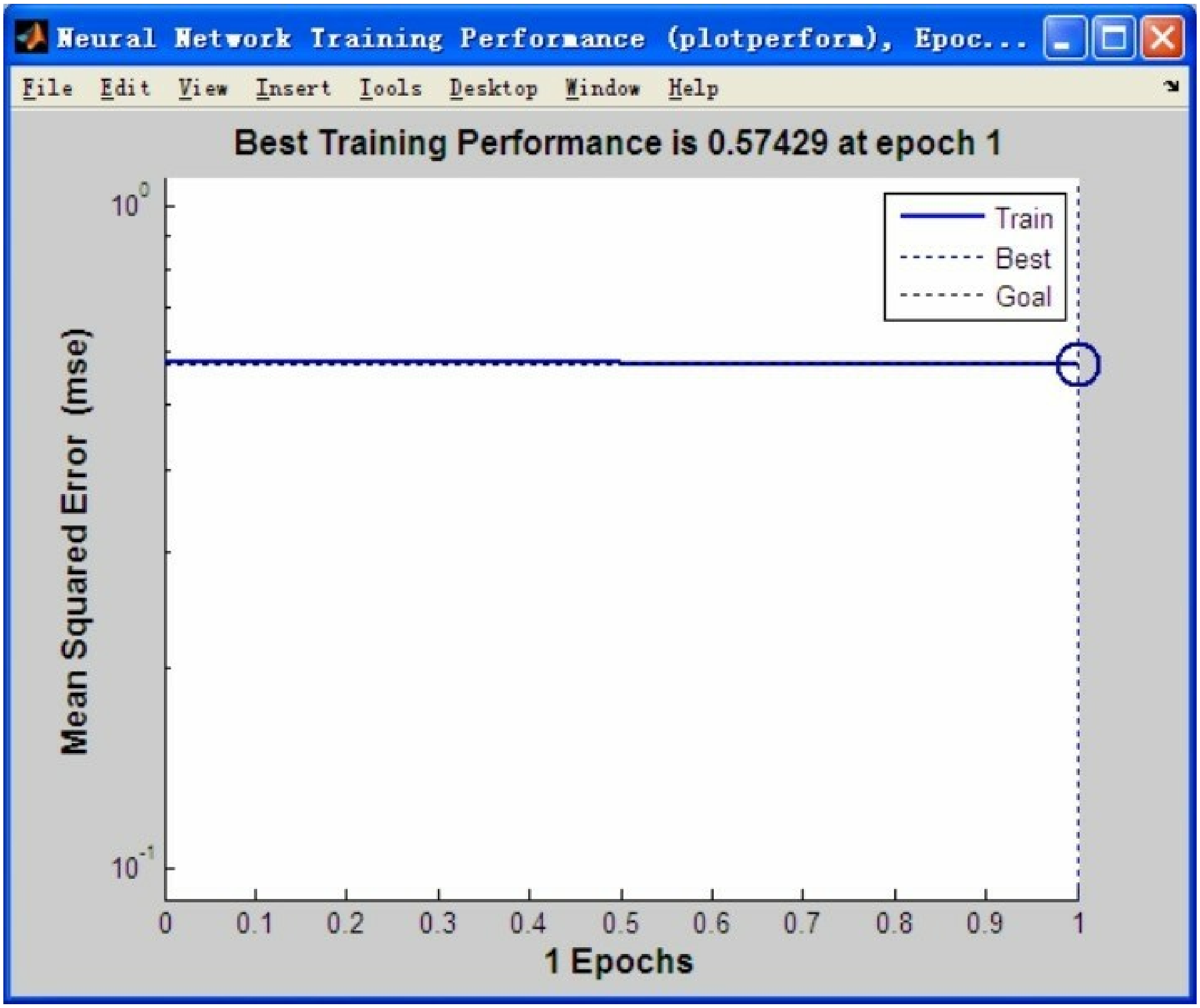


图6-5 网络训练误差曲线

由图6-5可以看出，网络经过1步训练就达到预期目标。

线性网络是一个包含dotprod权值函数、netsum网络输入函数和purelin传输函数的单独层网络。该层具有输入权值和阈值。权值和阈值通过initzero函数进行初始化。

线性层的适应或训练是用adaptwb或trainwb函数来完成的。adaptwb及trainwb函数用learnwh学习函数修正权值和阈值，用mse函数来计算线性网络的误差性能。

3.purelin

该函数是线性传输函数，可根据网络的输入计算线性层的输出。其

使用格式如下。

A=purelin(P)　%返回与输入向量对应的值

info=purelin(code)　%返回与每一个 code代码对应的有用信息

其中，P是网络的输入。

【例6-6】编写MATLAB代码，产生purelin的图形。

解：编写的MATLAB代码如下。

```
t=-2:0.1:2;
y=purelin(t)
plot(t,y)
```

运行代码得到结果。

```
y =
  Columns 1 through 10
    -2.0000    -1.9000    -1.8000    -1.7000    -1.6000    -1.5000    -1.400(
  Columns 11 through 20
    -1.0000    -0.9000    -0.8000    -0.7000    -0.6000    -0.5000    -0.400(
  Columns 21 through 30
    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000
0.7000    0.8000 0.9000
  Columns 31 through 40
    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000
1.7000    1.8000    1.9000
  Column 41
    2.0000
```

purelin传递函数如图6-6所示。

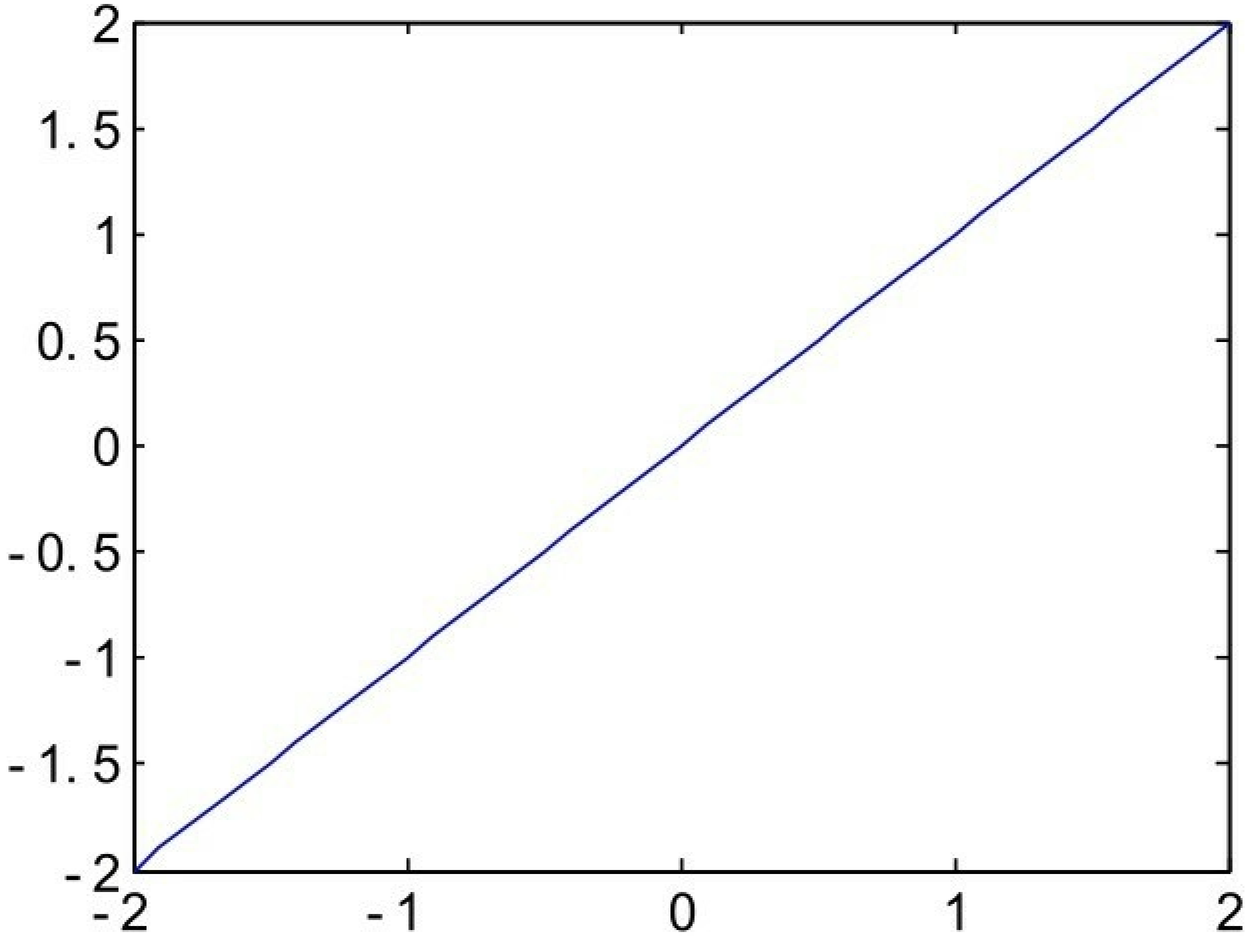


图6-6 purelin传递函数

一般在调用newlin或newlind函数生成一个标准的线性网络的过程中，都使用了purelin函数。用purelin函数改变网络的某一层时，需要进行如下设置。

NET.layers{i}.transferFcn=purelin

4.learnwh

该函数是LMS算法的学习函数。该函数可以修改神经元的权值和阈值。只要学习率不超出用maxlinlr函数计算的最大值，网络就可以收敛。其使用格式如下。

[dW,LS]=learnwh(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)

[db,LS]=learnwh(b,ones(1,Q),Z,N,A,T,E,gW,gA,D,LP,LS)

info=learnwh(code)

其中，W是权值矩阵。

P是输入向量。

Z是加权输入向量。

N为网络输入向量。

A为输出向量。

T为目标向量。

E为误差向量。

gW为误差性能梯度。

gA为误差性能输出地图。

LP为学习参数。

LS为学习状态。

dW为权值或阈值变化矩阵。

LS为新的学习状态。

【例6-7】给出一个具有3个输入、4个神经元的神经网络，对其定义一个随机输入P和误差E，给出其学习率lp =0.4，计算其权值变化矩阵。

解：首先随机产生输入向量和误差向量，程序如下。

```
P=rand(3,1);
E=rand(4,1);
```

根据上述输入和误差向量，计算权值变化矩阵。

```
lp.lr =0.4;
dw=learnwh([],P,[] ,[] ,[] ,[],E,[] ,[] ,[],lp,[])
```

程序仿真结果如下。

```
P =
   0.1419
   0.4218
   0.9157
```

E =

0.7922

0.9595

0.6557

0.0357

dw =

0.0450   0.1336   0.2902

0.0545   0.1619   0.3515

0.0372   0.1106   0.2402

0.0020   0.0060   0.0131

通过调用 learnwh 函数，可以产出一个标准的线性网络。按照 LMS 算法学习准则，learnwh函数从一个给定的神经元的输入P、误差E、权值或者阈值的学习率中，可以得到该神经元的权值变化。

$$dw = lr \cdot e \cdot pn'$$

5.maxlinlr

该函数的作用是计算线性网络的最大学习率。其调用格式如下。

lr=maxlinlr(P)   %返回一个不带阈值的线性网络所需要的最大学习率

lr=maxlinlr(P,’bias’)   %返回一个带有阈值的线性网络所需要的最大学习率

【例6-8】对于3 组三维输入向量P=[1 2 3;4 2 1;3 5 2]，求出其最大学习率。

解：使用函数maxlinlr进行求解，MATLAB代码如下。

```
P=[1 2 3;4 2 1;3 5 2];
lr=maxlinlr(P,'bias')
```

语句执行后得到的结果如下。

lr =

0.0149

6.errsurf

该函数的作用是计算单个神经元的误差曲面。其调用格式如下。

e=errsurf(p,t,wv,bv,f)

其中，P为输入向量。

t为单目标向量。

wv为权值行向量。

bv为阈值B的行向量。

f为传递函数，且在wv和bv上返回误差矩阵。

【例6-9】随机给出一个一维的单个神经元的输入和目标向量，当权值从−1到1、阈值从−3到3变化时，计算单个purelin神经元的误差曲面。

解：首先随机产生输入向量和误差向量，程序如下。

```
P=[-3 -2.2 -4.5 -3 3 3.1 1 5.1];
T=[1 0 2.5 1 0.01 0.04 3 1];
```

根据上述输入和目标向量，计算神经元的误差曲面。

```
wv=-1:0.1:1;
bv=-3:0.3:3;
e=errsurf(P,T,wv,bv,’purelin’)
plotes(wv,bv,e,[60 30])
```

得到的误差曲面如图6-7所示。

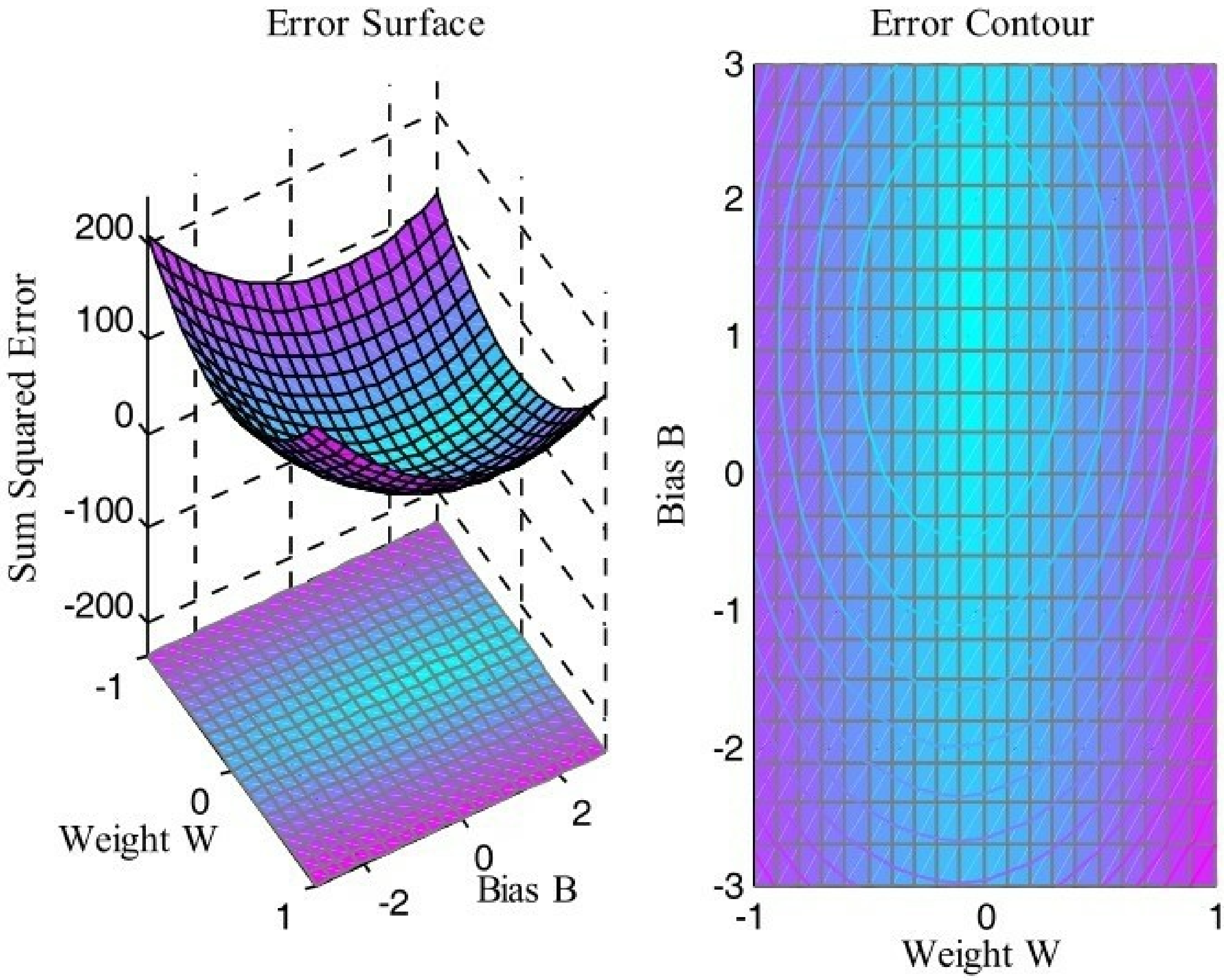

Error Surface
Sum Squared Error
200
100
0
-100
-200
-1
0
Weight W
1
-2
0
Bias B
2
Error Contour
3
2
1
0
-1
-2
-3
Bias B
-1
0
1
Weight W

图6-7 误差曲面

6.3 线性神经网络的MATLAB仿真程序设计

6.3.1 线性神经网络设计的基本方法

除了创建神经网络的函数不同，线性神经网络与感知器神经网络在程序设计方法上基本相同。此外，线性神经网络还可以用设计函数创建，且不需要进行训练。线性神经网络的MATLAB仿真程序设计分为以下几个步骤。

（1）创建神经网络。首先根据所需要解决的问题，确定输入向量的维数、取值范围、神经元个数等，然后根据函数newlin来创建线性神经网络。

（2）训练神经网络。首先需要确定训练网络的样本集、每个样本的输入向量和目标向量，然后使用函数train对网络进行训练或者使用adapt自适应调整网络的权值和阈值。根据训练的结果，决定是否调整训练参数，从而得到满足误差性能指标的神经网络。

注意：如果使用newlind函数设计线性神经网络，则网络不需要进行训练。

（3）网络仿真。训练完神经网络后，需要构造测试神经网络的样本集，加载训练好的神经网络后，使用sim函数对网络进行仿真，并保持网络的仿真结果。

在有些神经网络的应用中，自适应神经网络的输出不是取自线性神经元的输出，而是目标响应t与模拟输出量a的误差e（e=t−a）。

从以上过程中可以看出，线性神经网络设计中需要用到newlin、newlind、train、adapt和sim函数。

6.3.2 线性神经网络的设计

在模式识别、函数逼近和信号滤波等方面，线性神经网络有着广泛的应用。下面通过几个经典案例，详细说明线性神经网络在MATLAB中的仿真程序设计。

1.模式分类

【例6-10】以单层线性网络模拟与函数。

解：与函数的真值表见表6-2。

表6-2 与函数真值表

输入 P_1	输入 P_2	输出 y
0	0	0
1	0	0
0	1	0
1	1	1

如果将与函数看作 P_1和 P_2平面上的点，则点 $A_0(0,0)$，$A_1(0,1)$，$A_2(1,0)$表示输出为 0的3个点，$A_3(1,1)$表示输出为1的一个点，与函数的图形表示如图6-8所示。

从图 6-8 可以看出，与函数是一个简单的线性划分问题，用一个线性神经元网络就可以实现其功能。

根据上述分析，设计线性神经网络具有1个输入向量，包括两个元素，输入元素的取值范围为[0 1]，输出向量也是 1 个元素，为二值变量 1 或 0。设计好的线性神经网络结构示意图如图6-9所示。

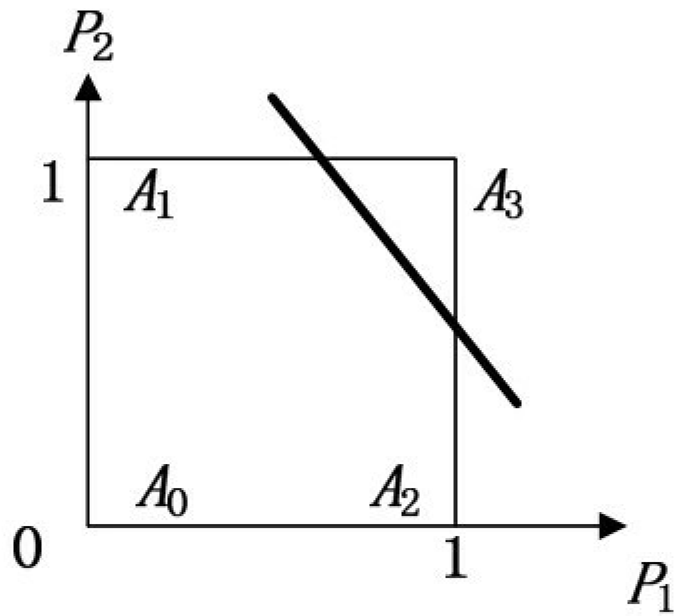


图6-8 与函数的图形表示

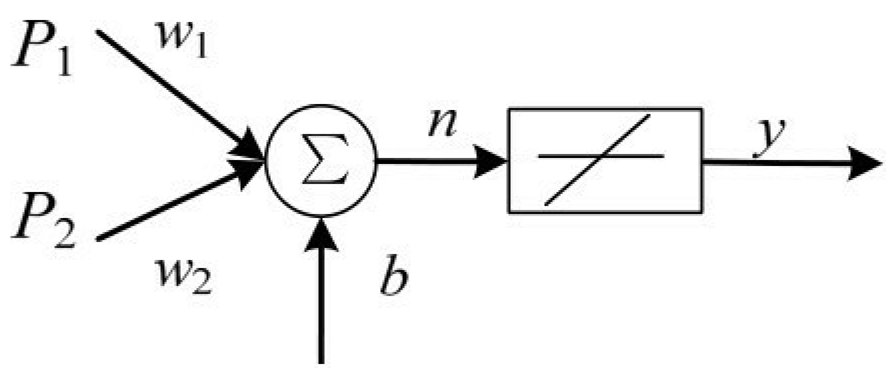


图6-9 设计好的线性神经网络结构示意图

单层线性网络实现与函数的MATLAB仿真程序代码如下所示。

```
clear all
clc
%%%%%%%%创建和设计一个神经网络%%%%%%%%%%%%%%%
P=[0 0;0 1;1 0;1 1]';
T=[0 0 0 1];
net=newlind(P,T);    %设计线性神经网络
```

```
w=net.IW{1}       %输出训练后的权值
b=net.b{1}       %输出训练后的阈值
%%%%%%%%线性神经网络仿真%%%%%%%%%%%%%%%%%%%%
n=sim(net,P)       %输出仿真结果
y=n>0.6          %将模拟仿真结果转换为数字量
```

代码运行结果如下。

```
w =
   0.5000    0.5000
b =
   -0.2500
n =
   -0.2500    0.2500    0.2500    0.7500
y =
   0    0    0    1
```

由以上运行结果可知，当输出是模拟量n时，结果与目标之间的误差为0.25，如果将结果y与门限值（本题设定门限值为0.6）比较，即可消除误差。在程序中，使用语句y=n>0.6来实现消除误差。

2.噪声对消

【例6-11】以自适应线性网络实现噪声对消。

解：对于一般的滤波器，很难通过滤波器将信号中的噪声消除，但是对于一个最优的滤波器，可以使用自适应线性网络来实现噪声对消，其原理框图如图6-10所示。

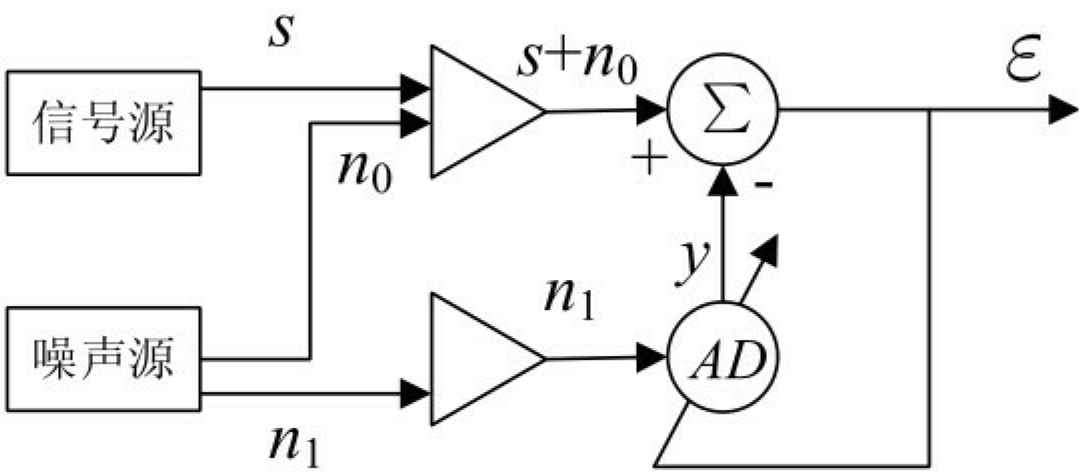


图6-10 噪声对消原理图

图6-10中s以设置为平稳的零均值随机信号作为原始输入信号；n_0为与s不相关的随机噪声；n_1为与n_0相关的信号；系统输出为ε；y为神经元的输出。

$$\varepsilon = s + n_0 - y$$

$$E[\varepsilon^2] = E[(s + n_0 - y)^2] = E[s^2] + 2E[s \cdot (n_0 - y)] + E[(n_0 - y)^2] = E[s^2] + E[(n_0 - y)^2]$$

通过神经元的调节，可以得到

$$E_{\min}[\varepsilon^2] = E_{\min}[s^2] + E_{\min}[(n_0 - y)^2]$$

在上式中，当$E_{min}[(n_0-y)^2]\rightarrow 0$时，$y\rightarrow n_0$，其输出ε为s，则噪声被抵消。

采用这种系统来完成对胎儿心率的检测，可以得到十分满意的结果。由于测量胎儿的心率一定会受到母体心率的干扰，而且目前心率很强，但与胎儿心率是相互独立的，所以可以将母体心率作为噪声源n_1输入到神经元中，混有噪声的胎儿心率信号作为目标响应，通过对消后，系统就可以得到清晰的胎儿心率。

这种系统还可以用于电话中的回音对消。如果在电话通话中没有回音对消措施，那么对方的声音会淹没在自身的声音中，严重影响通话质

量。如果将自身的声音作为噪声源n_1输入到神经元中，混有对方声音的信号作为目标响应，系统通过对消后可以得到对方的清晰的声音信号。

假设传输信号为余弦波信号，噪声为随机噪声，进行自适应线性神经网络设计。

根据以上分析，自适应线性神经元的输入向量为随机噪声n_1；余弦波信号与随机噪声之和为神经元的目标向量；输出信号为网络调整过程中的误差信号。

实现噪声对消的MATLAB程序如下。

```
clear all
clc
%%%%%%%%定义输入向量和目标向量%%%%%%%%%%%%%%
t=0.01:0.02:20;       %时间变量
zs=(rand(1,1000)-0.5)*10;    %随机噪声
input=cos(t);
P=zs;                %输入向量
T=input+zs;           %目标向量
%%%%%%%%%%创建线性神经网络%%%%%%%%%%%%%%%%
net=newlin([-1 1],1,0,0.0005);
%%%%%%%%%%%神经网络自适应训练%%%%%%%%%%%%%
net.adaptParam.passes=70;
[net,y,output]=adapt(net,P,T)    %输出信号output为网络调整过程中的误差
%%%%%%%%%%%%%%绘制信号波形%%%%%%%%%%%%%%%%%%
```

```
hold on
subplot(3,1,1);
plot(t,input)
xlabel('T','position',[12,-2]);
ylabel('信号波形cos(t)','fontsize',8)
subplot(3,1,2);
plot(t,T);
xlabel('T','position',[12,-4]);
ylabel('随机噪声波形cos(t)+zs','fontsize',8)
subplot(3,1,3);
plot(t,output);
xlabel('T','position',[12,-1]);
ylabel('输出信号波形y(t)','fontsize',8)
hold off
```

运行上述程序，得到图6-11所示结果。从该图中可以得到，除了含有的直流分量，输出信号波形与输入信号波形基本一致，即该线性神经网络消除了随机噪声。

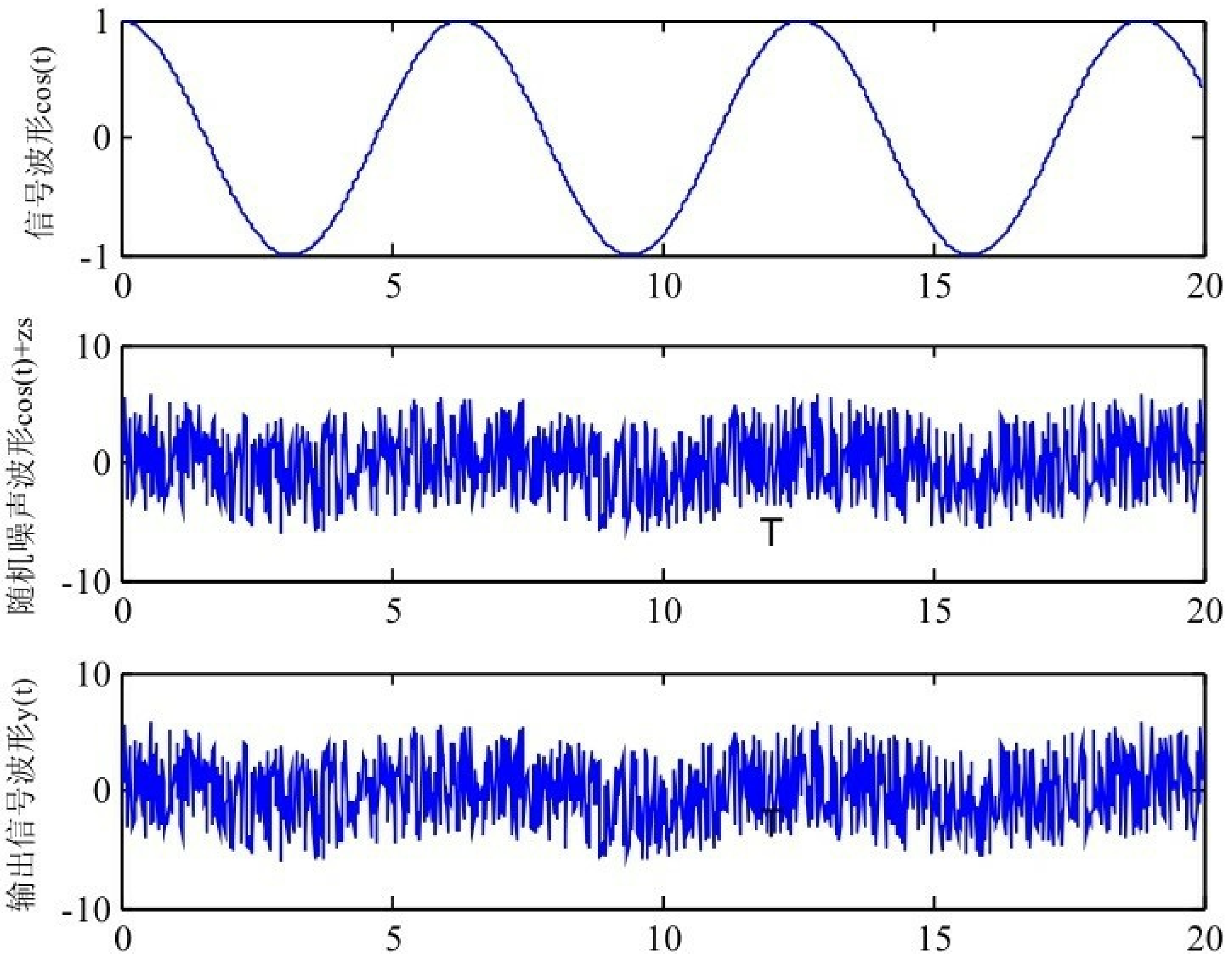


图6-11 噪声消除程序运行结果

3.信号预测

【例6-12】假如有一个自适应滤波器结构如图6-12所示，该滤波器的目的是从输入信号的前两个时刻的值预测当前时刻的值。图中D为延迟单元，多个延迟单元可以构成抽头延迟，如图6-13所示。

设输入信号为一个随机序列，试编写MATLAB程序，画出符合上述要求的自适应滤波器的输入和输出波形。

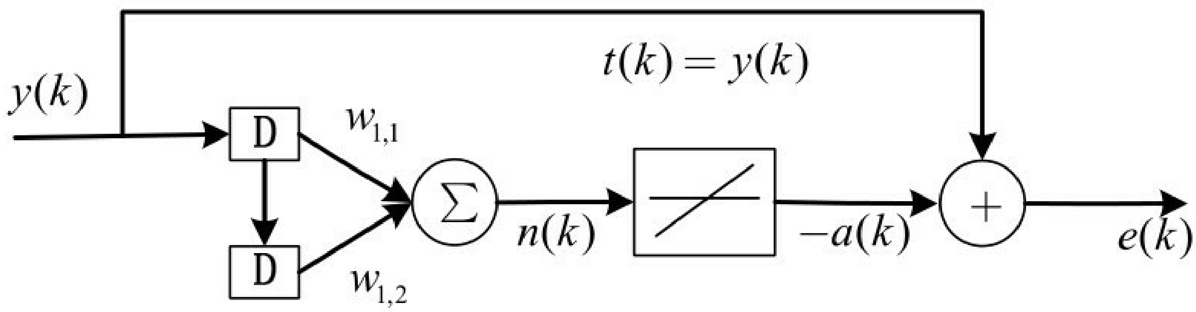


图 6-12　自适应滤波器结构图

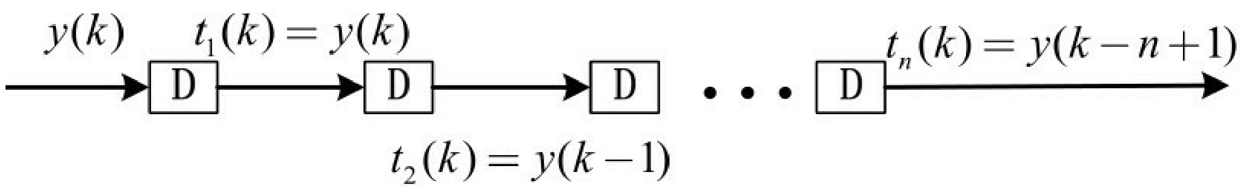


图6-13 由多个延迟单元构成的抽头延迟结构

解：线性神经网络是使用串行方式按照时间顺序输入的，其MATLAB程序代码如下所示。

```
clear all
clc
%%%%%%%%定义输入向量和目标向量%%%%%%%%%%%%
time=0.1:0.1:20;        %时间变量
y=(rand(1,200)-0.6)*5;        %定义随机输入信号
P=con2seq(y);
delays=[12];            %神经元输入延迟量
T=P;                %神经元的目标向量
%%%%%%%%创建线性神经网络%%%%%%%%%%%%%%%%%%
net=newlin(minmax(y),1,delays,0.0005);
%%%%%%%%线性神经网络的自适应训
```

```
练%%%%%%%%%%%
    net.adaptParam.passes=70;
    [net,a,output]=adapt(net,P,T)
    %%%%%%%%%绘制波
形%%%%%%%%%%%%%%%%%%%%%%%%%%%
    hold on
    subplot(3,1,1)
    plot(time,y)            %随机输入信号波形
    xlabel('T','position',[20,-2]);
    ylabel('随机输入信号s(t)')
    axis([0 20 -3 3])
    subplot(3,1,2);
    output=seq2con(output);
    plot(time,output{1});        %预测输出信号波形
    xlabel('T','position',[20,-2]);
    ylabel('预测输出信号y(t)')
    axis([0 20 -3 3])
    subplot(3,1,3);
    e=output{1}-y;
    plot(time,e);            %误差曲线
    xlabel('time','position',[20,-2]);
    ylabel('误差曲线e(t)')
    axis([0 20 -0.2 0.2])
    hold off
```

其运行结果如图6-14所示。

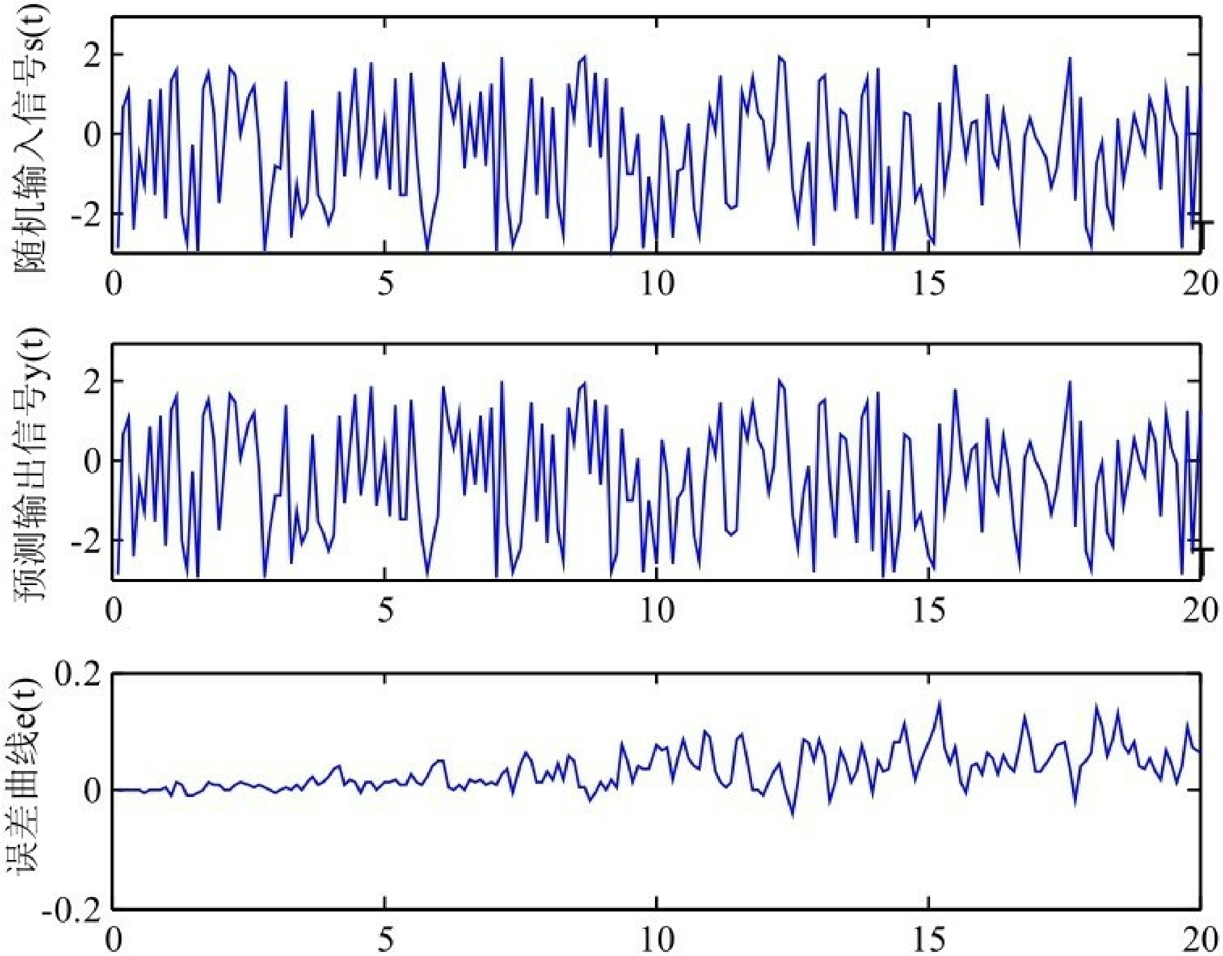


图6-14 程序运行结果

从图6-14中可以看出，输出信号波形与输入信号波形误差较小，图形基本相同，由此可知，该网络的输出很好地预测了输入，符合题中所提要求。

注意：在程序设计中，需要注意学习率和训练步长的选择。学习率过大，学习过程会不稳定；学习率过小，学习过程将变长，需要的训练步长数增大。如果选择不当，会得到不满意的结果。

6.4 本章小结

本章介绍了线性神经网络的模型，详细讨论了线性神经网络的LMS学习规则。线性神经网络广泛用于自适应滤波方面，所以正确理解和掌握线性神经网络模型及其学习规则非常重要。线性神经网络工具箱函数有很多，本章重点介绍了在MATLAB程序设计中经常用到的几个工具箱函数。最后还介绍了神经网络的MATLAB程序仿真设计方法及其应用。

第7章 BP神经网络

BP（Back Propagation）网络是1986年由Rumelhart和McCelland 为首的科学家小组提出的，是一种按误差逆传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型之一。

BP神经网络算法是在BP神经网络现有算法的基础上提出的，是通过任意选定一组权值，将给定的目标输出直接作为线性方程的代数和来建立线性方程组，解得待求权，不存在传统方法的局部极小及收敛速度慢的问题，且更易理解。

本章将主要介绍BP神经网络原理、学习算法、工具箱函数及工程应用。

学习目标：

·熟悉BP神经网络原理

·了解BP学习算法

·掌握BP神经网络工具箱函数的使用

·掌握BP神经网络的工程应用

7.1 BP神经网络原理

BP网络能学习和存储大量的输入—输出模式映射关系，而无需事前揭示描述这种映射关系的数学方程。它的学习规则是使用最速下降法，通过反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。BP　神经网络模型拓扑结构包括输入层（input）、隐含层（hide layer）和输出层（output layer）。

7.1.1 BP 神经网络模型

BP神经元模型如图7-1所示。

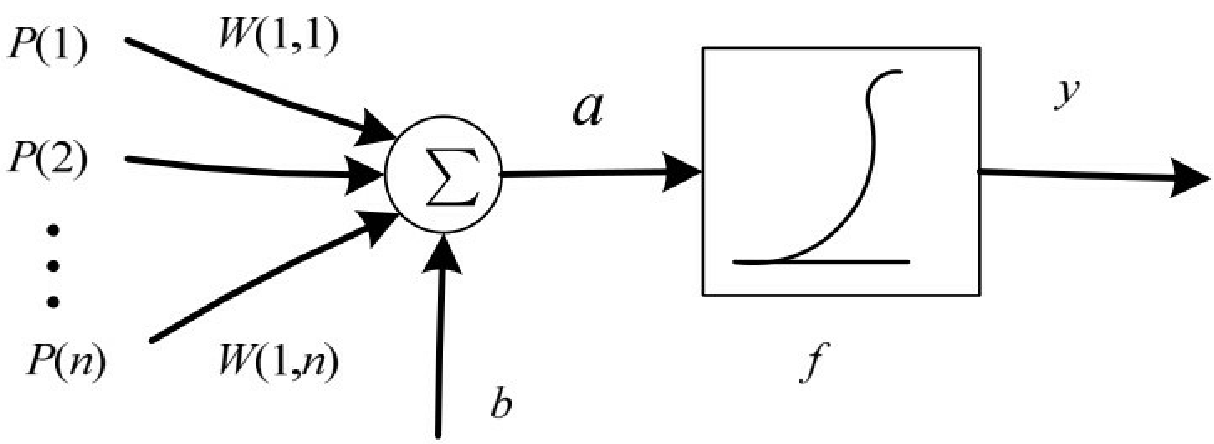


图7-1 BP神经元模型

BP神经元的输入为P，权值和阈值分别为w和b，线性神经元模型的输出为y。BP神经元与其他神经元类似，不同的是 BP 神经元的传递函数为非线性函数，最常用的传递函数为logsig和tansig函数，如图6-2所示。有时候BP神经网络的传递函数也采用purelin。

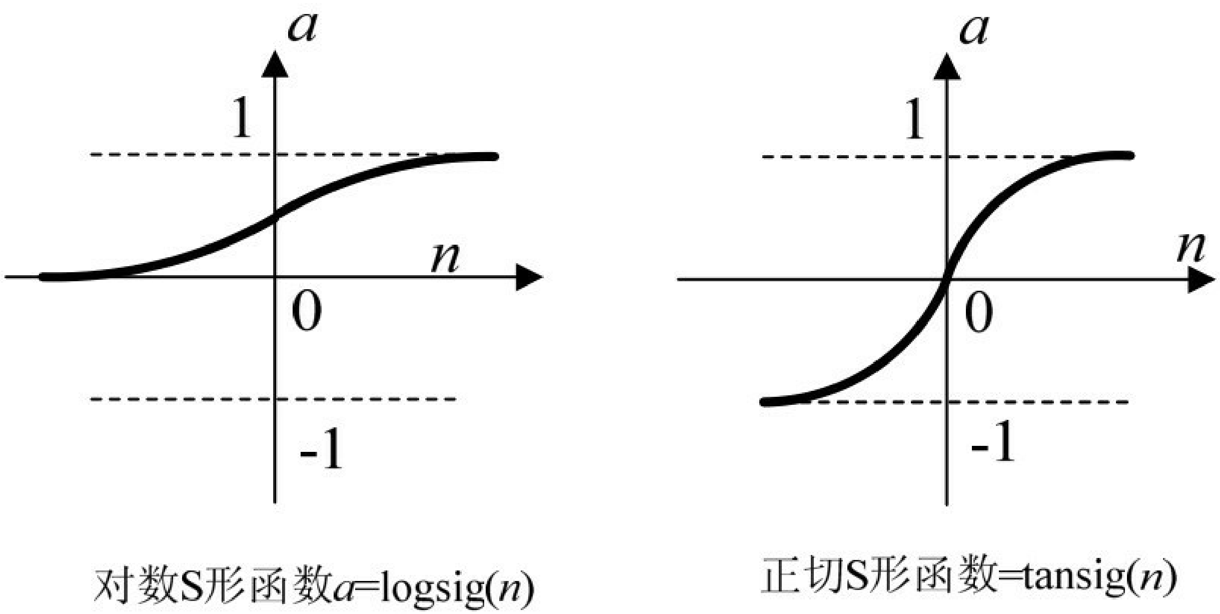


图7-2 BP神经网络常用的传递函数

BP网络是一种多层前馈神经网络，由输入层、隐含层和输出层组成。图7-3为一个典型的三层 BP 神经网络的结构图，层与层之间采用全互连方式，同一层之间不存在相互连接，隐含层可以有一层或多层。

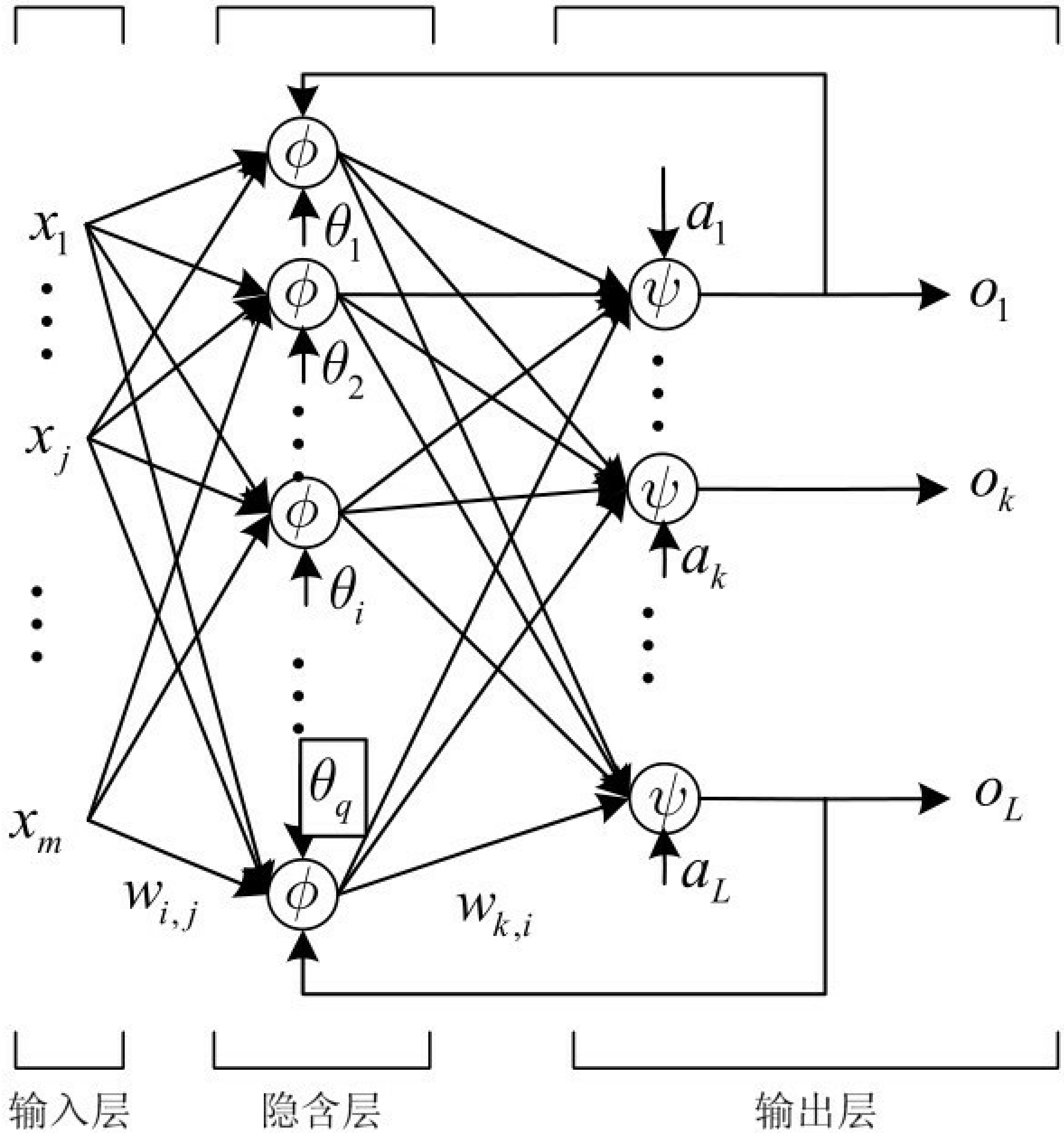


图7-3 典型三层BP神经网络结构图

其中， x_j表示输入层第j个节点的输入，j=1,…,M。

$w_{i,j}$表示隐含层第i 个节点到输入层第j 个节点之间的权值。

$_i\theta$表示隐含层第i个节点的阈值；

φ表示隐含层的激励函数；

$w_{k,i}$表示输出层第k　　个节点到隐含层第i个节点之间的权值，i=1,

…,q。

a_k表示输出层第k 个节点的阈值，k=1,…,L。

ψ表示输出层的激励函数。

o_k表示输出层第k 个节点的输出。

层与层之间有两种信号在流通。一种是工作信号（用实线表示），它是施加输入信号后向前传播直到在输出端产生实际输出的信号，是输入和权值的函数。另一种是误差信号（用虚线表示），网络实际输出与期望输出间的差值即为误差，它由输出端开始逐层向后传播。

BP网络的学习过程由前向计算过程和误差反向传播过程组成。在前向计算过程中，输入量从输入层经隐含层逐层计算，并传向输出层，每层神经元的状态只影响下一层神经元的状态。如输出层不能得到期望的输出，则转入误差反向传播过程，误差信号沿原来的连接通路返回，逐次调整网络各层的权值和阈值，直至到达输入层，再重复向计算。

这两个过程一次反复进行，不断调整各层的权值和阈值，使得网络误差最小或达到人们所期望的要求时，学习过程结束。

7.1.2 BP神经网络算法

基本 BP 神经网络算法包括两个方面：信号的前向传播和误差的反向传播。即计算实际输出时按从输入到输出的方向进行，而权值和阈值的修正从输出到输入的方向进行。根据图7-3所示参数，BP神经网络算法的计算过程如下所示。

（1）信号的前向传播过程。

隐含层第i个节点的输入net_i为

$$net_i = \sum_{j=1}^{M} w_{ij} x_j + \theta_i$$

隐含层第i个节点的输出o_i为

$$o_i = \phi(net_i) = \phi(\sum_{j=1}^{M} w_{ij} x_j + \theta_i)$$

输出层第k个节点的输入net_k为

$$net_k = \sum_{i=1}^{q} w_{ki} y_i + a_k = \sum_{i=1}^{q} w_{ki} \phi(\sum_{j=1}^{M} w_{ij} x_j + \theta_i) + a_k$$

输出层第k个节点的输出o_k为

$$o_k = \psi(net_k) = \psi(\sum_{i=1}^{q} w_{ki} y_i + a_k) = \psi[\sum_{i=1}^{q} w_{ki} \phi(\sum_{j=1}^{M} w_{ij} x_j + \theta_i) + a_k]$$

（2）误差的反向传播过程。

误差的反向传播，即首先由输出层开始逐层计算各层神经元的输出误差，然后根据误差梯度下降法来调节各层的权值和阈值，使修改后的网络的最终输出能接近期望值。

对于每一个样本p的二次型误差准则函数为Ep

$$E_p = \frac{1}{2} \sum_{k=1}^{L} (T_k - o_k)^2$$

系统对P个训练样本的总误差准则函数为

$$E_p = \frac{1}{2} \sum_{p=1}^{P} \sum_{k=1}^{L} (T^p{}_k - o^p{}_k)^2$$

根据误差梯度下降法依次修正输出层权值的修正量Δw_{ki}，输出层阈值的修正量Δa，隐含层权值的修正量Δw_{ij}，隐含层阈值的修正量$\Delta \theta_i$。

$$\Delta w_{ki} = -\eta \frac{\partial E}{\partial w_{ki}}; \Delta a_k = -\eta \frac{\partial E}{\partial a_k}; \Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}; \Delta \theta_i = --\eta \frac{\partial E}{\partial \theta_i}$$

输出层权值调整公式

$$\Delta w_{ki} = -\eta \frac{\partial E}{\partial w_{ki}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{ki}} = -\eta \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial w_{ki}}$$

输出层阈值调整公式

$$\Delta a_k = -\eta \frac{\partial E}{\partial a_k} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial a_k} = -\eta \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial a_k}$$

隐含层权值调整公式

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}}$$

隐含层阈值调整公式

$$\Delta \theta_i = -\eta \frac{\partial E}{\partial \theta_i} = -\eta \frac{\partial E}{\partial net_i} \frac{\partial net_i}{\partial \theta_i} = -\eta \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial net_i} \frac{\partial net_i}{\partial \theta_i}$$

又因为

$$\frac{\partial E}{\partial o_k} = -\sum_{p=1}^{P} \sum_{k=1}^{L} (T_k{}^p - o_k{}^p)$$

$$\frac{\partial net_k}{\partial w_{ki}} = y_i; \frac{\partial net_k}{\partial a_k} = 1; \frac{\partial net_i}{\partial w_{ij}} = x_j; \frac{\partial net_i}{\partial \theta_i} = 1;$$

$$\frac{\partial E}{\partial y_i} = -\sum_{p=1}^{P}\sum_{k=1}^{L}(T^p{}_k - o^p{}_k)\bullet\psi^{'}(net_k)\bullet w_{ki}$$

$$\frac{\partial y_i}{\partial net_i} = \phi^{'}(net_i);$$

$$\frac{\partial o_k}{\partial net_k} = \psi^{'}(net_k);$$

所以最后得到以下公式

$$\Delta w_{ki} = \eta\sum_{p=1}^{P}\sum_{k=1}^{L}(T^p{}_k - o^p{}_k)\bullet\psi^{'}(net_k)\bullet y_i$$

$$\Delta a_k = \eta\sum_{p=1}^{P}\sum_{k=1}^{L}(T^p{}_k - o^p{}_k)\bullet\psi^{'}(net_k)$$

$$\Delta w_{ij} = \eta\sum_{p=1}^{P}\sum_{k=1}^{L}(T^p{}_k - o^p{}_k)\bullet\psi^{'}(net_k)\bullet w_{ki}\bullet\phi^{'}(net_i)\bullet x_j$$

$$\Delta\theta_i = \eta\sum_{p=1}^{P}\sum_{k=1}^{L}(T^p{}_k - o^p{}_k)\bullet\psi^{'}(net_k)\bullet w_{ki}\bullet\phi^{'}(net_i)$$

BP神经网络算法程序流程图如图7-4所示。

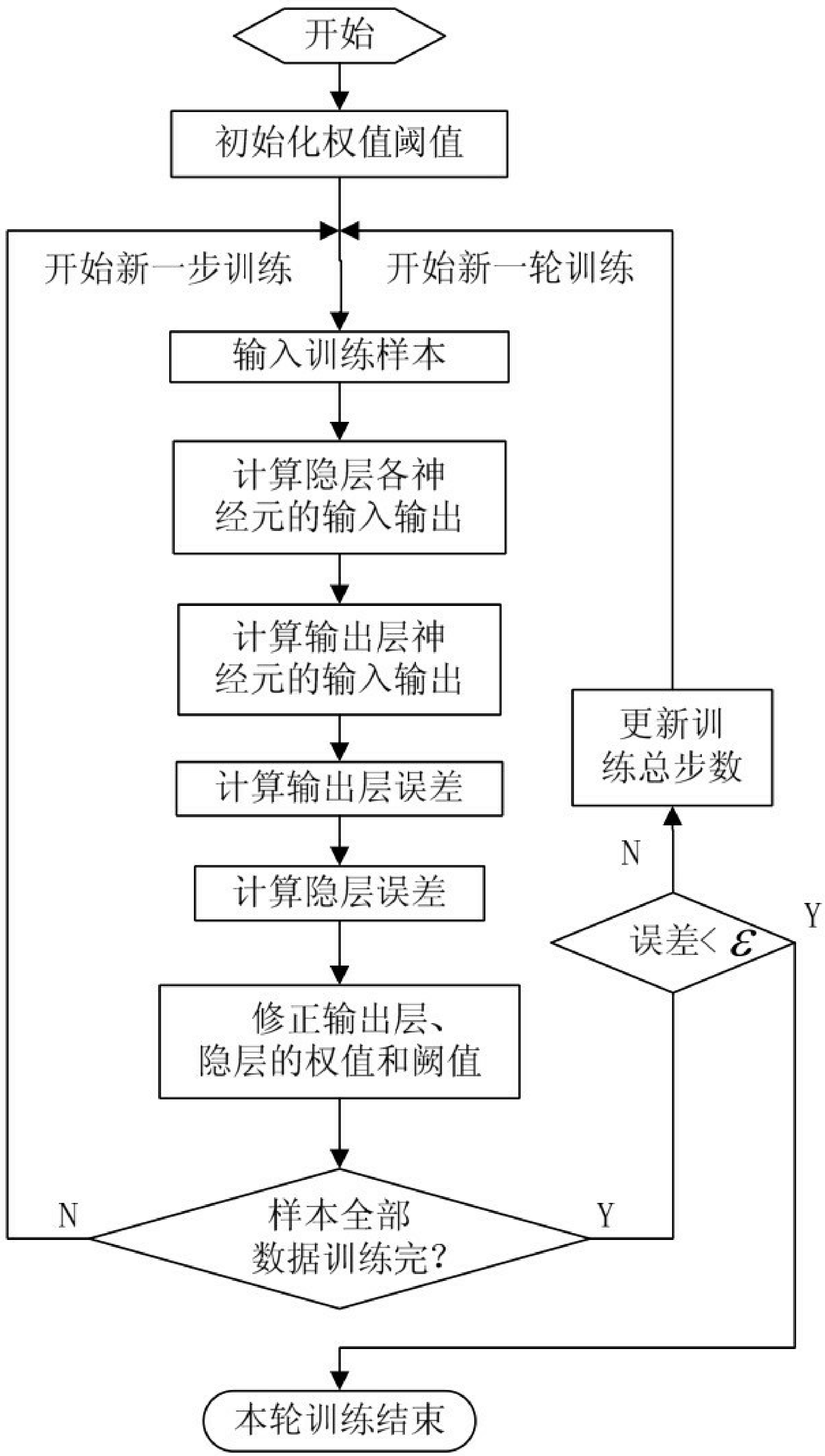

开始
初始化权值阈值
开始新一步训练
开始新一轮训练
输入训练样本
计算隐层各神经元的输入输出
计算输出层神经元的输入输出
计算输出层误差
计算隐层误差
修正输出层、隐层的权值和阈值
样本全部数据训练完?
N
Y
误差< ε
N
Y
更新训练总步数
本轮训练结束

图7-4 BP算法程序流程图

BP算法因其简单、易行、计算量小、并行性强等优点，目前是神经网络训练采用最多也是最成熟的训练算法之一。其算法的实质是求解误差函数的最小值问题，由于它采用非线性规划中的最速下降方法，按误差函数的负梯度方向修改权值，因而通常存在以下问题。

·学习效率低，收敛速度慢。

·易陷入局部极小状态。

针对以上问题，一般常用三种方式对BP算法进行改进。

（1）附加动量法。

附加动量法使网络在修正其权值时，不仅考虑误差在梯度上的作用，而且考虑在误差曲面上变化趋势的影响。在没有附加动量的作用下，网络可能陷入浅的局部极小值，利用附加动量的作用有可能滑过这些极小值。

该方法是在反向传播法的基础上在每一个权值（或阈值）的变化上加上一项正比于前次权值（或阈值）变化量的值，并根据反向传播法来产生新的权值（或阈值）变化。

带有附加动量因子的权值和阈值调节公式为

$$\Delta w_{ij}(k+1) = (1-mc)\eta\delta_i p_j + mc\Delta w_{ij}(k)$$

$$\Delta b_i(k+1) = (1-mc)\eta\delta_i + mc\Delta b_i(k)$$

其中k为训练次数，mc为动量因子，一般取0.95左右。

附加动量法的实质是将最后一次权值（或阈值）变化的影响，通过一个动量因子来传递。当动量因子取值为零时，权值（或阈值）的变化仅根据梯度下降法产生。

当动量因子取值为1时，新权值（或阈值）的变化则是设置为最后一次权值（或阈值）的变化，依梯度法产生的变化部分则被忽略。

当增加了动量项后，促使权值的调节向着误差曲面底部的平均方向变化。当网络权值进入误差曲面底部的平坦区时，δ_i将变得很小，这时$\Delta w_{ij}(k+1)=\Delta w_{ij}(k)$。而$\Delta w_{ij}=0$的出现有助于使网络从误差曲面的局部极小值中跳出。

根据附加动量法的设计原则，当修正的权值在误差中导致太大的增长结果时，新的权值应被取消而不被采用，同时动量作用停止下来，以使网络不进入较大误差曲面；当新的误差变化率对其旧值超过一个事先设定的最大误差变化率时，也得取消所计算的权值变化。其最大误差变化率可以是任何大于或等于1的值。通常取值1.04。所以，在进行附加动量法的训练程序设计时，必须加进条件判断以正确使用其权值修正公式。

训练程序设计中采用动量法的判断条件为

$$mc=\begin{cases}0, E(k)>E(k-1)\cdot 1.04\\ 0.95, E(k)<E(k-1)\\ mc,\text{其他}\end{cases}，\quad E(k)\text{为第}k\text{步误差平方和}$$

（2）自适应学习速率。

对于一个特定的问题，学习速率通常是凭经验或实验获取，但即使这样，对训练开始初期功效较好的学习速率，不一定对后来的训练合适。为了解决这个问题，需要在训练过程中自动调节学习速率。

通常调节学习速率的准则是检查权值是否真正降低了误差函数，如果确实如此，则说明所选学习速率小了，可以适当增加一个量；若不是这样，而产生了过调，那么就应该减少学习速率的值。下式给出了一个自适应学习速率的调整公式。

$$\eta(k+1)=\begin{cases}1.05\eta(k), E(k+1)<E(k)\\0.7\eta(k), E(k+1)>1.04E(k)\\\eta(k),\text{其他}\end{cases}$$，$E(k)$ 为第 k 步误差平方和

初始学习速率η(0)的选取范围可以有很大的随意性。

（3）动量—自适应学习速率调整算法。

当采用前述的动量法时，BP算法可以找到全局最优解，而当采用自适应学习速率时， BP算法可以缩短训练时间，这两种方法也可以用来训练神经网络，该方法称为动量—自适应学习速率调整算法。

7.1.3 BP神经网络的训练

这一节描述了两种不同的训练方式：增加方式和批处理方式。

在增加方式中，每提交一次输入数据，网络权重和偏置都更新一次。在批处理方式中，仅仅当所有的输入数据都被提交以后，网络权重和偏置才被更新。

1.增加方式（应用与自适应网络和其他网络）

虽然增加方式更普遍地应用于动态网络，比如自适应滤波，但实际上在静态和动态网络中都可以应用它。

（1）静态网络中的增加方式。

用增加方式来训练网络，每提交一次输入数据，网络权重和偏置都更新一次。假定神经网络的输入如下。

p=[1 2;2 1;2 3;3 1];

目标输出如下。

t=[4 5 7 7];

建立线性函数：$t=2p_1+p_2$。

首先用 0 初始化权重和偏置。为了显示增加方式的效果，学习速率也设为 0。具体程序如下所示。

net = newlin([-1 1;-1 1],1,0,0);

net.IW{1,1}= [0 0];

net.b{1}= 0;

为了用增加方式，把输入向量和目标向量按照以下方式表示。

P ={[1;2] [2;1] [2;3] [3;1]};

T ={4 5 7 7};

通常情况下，不论是作为一个同步向量矩阵输入还是作为一个异步向量细胞数组输入，模拟的输出值是一样的。但是在训练网络时，模拟的输出值会出现不一样。

当使用 adapt 函数训练神经网络时，如果输入是异步向量，那么权重将在每一组输入提交的时候更新（就是增加方式）；如果输入是同步向量，那么权重将只在所有输入提交的时候更新（就是批处理方式）。

下面开始用增加方式训练网络。

[net,a,e,pf] = adapt(net,P,T);

由于学习速度为0，网络输出也为0，并且权重没有被更新。误差值和目标值无变化。

a = [0] [0] [0] [0]

e = [4] [5] [7] [7]

如果设置学习速度为0.1，那么当每一组输入提交时，网络输出结果就会发生变化。

net.inputWeights{1,1}.learnParam.lr=0.1;

net.biases{1,1}.learnParam.lr=0.1;

[net,a,e,pf] = adapt(net,P,T);

训练得到如下结果。

a = [0 2 6.0 5.8]

e = [4 3 1.0 1.2]

由于在第一个输入数据提交前还没有更新，此时第一个输出值和学

习速率为0时的第一个输出值相同。当运行到第二步时，权重更新，第二个输出就会发生变化。每计算一次误差，权重都不断地修改。如果网络可行并且学习速率设置得当，误差将不断地趋向于0。

（2）动态网络中的增加方式。

训练动态网络同样也能用增加方式。根据以下程序建立一个神经网络，使初始化权重为0，并把学习速率设为0.1。

```
net = newlin([-1 1],1,[0 1],0.1);
net.IW{1,1}= [0 0];
net.biasConnect = 0;
```

为了用增加方式，将输入向量和目标输出表示如下。

```
Pi ={1};
P ={2 3 4};
T ={3 5 7};
```

用adapt来训练网络。

```
[net,a,e,pf] = adapt(net,P,T,Pi);
```

训练得到如下结果。

```
a = [0 2.4 7.98]
e = [3 2.6 -1.98]
```

由于权重没有更新，第一个输出是0，但后续每一个序列步进，权重都跟着改变一次。

2．批处理方式

在批处理方式中，仅仅当所有的输入数据都被提交以后，网络权重和偏置才被更新，它也可以应用于静态和动态网络。

（1）静态网络中的批处理方式。

批处理方式可以用adapt或train函数来实现，由于采用了更高效的学习算法，train通常是最好的选择。增加方式只能用adapt来实现，而train函数只能用于批处理方式。

参照增加方式静态网络中使用的网络，学习速率设置为0.1。

```
net = newlin([-1 1;-1 1],1,0,0.1);
net.IW{1,1}= [0 0];
net.b{1}= 0;
```

用adapt函数实现静态网络的批处理方式，输入向量必须用同步向量矩阵的方式放置。

```
P = [1 2 2 3; 2 1 3 1];
T = [4 5 7 7];
```

调用adapt函数将触发adaptwb函数，这是默认的线性网络调整函数。learnwh是默认的权重和偏置学习函数。因此，Widrow-Hoff学习法将会被使用，MATLAB代码如下。

```
[net,a,e,pf] = adapt(net,P,T);
```

训练得到如下结果。

```
>> a
a =
   0    0    0    0
>> e
e =
   4    5    7    7
```

因为在所有要训练的数据提交前权重没有被更新，所以网络的输出全部为 0。如果在程序中显示权重，就会有如下结果。

```
>> net.IW{1,1}
ans =
   4.9000    4.1000
>> net.b{1}
ans =
   2.3000
```

现在用train函数来实现批处理方式。由于Widrow-Hoff规则能够在增加方式和批处理方式中应用，它可以通过adapt和train触发。我们有好几种算法只能用于批处理方式（特别是Levenberg-Marquardt算法），所以这些算法只能用train触发。

用train函数，任何异步向量细胞数组都会转换成同步向量矩阵。这是因为网络是静态的，并且因为train总是在批处理方式中使用。因为MATLAB实现同步模式效率更高，所以只要可能，总是采用同步模式处理。

（2）动态网络中的增加方式。

训练静态网络相对要简单一些。如果用train函数训练网络，即使输入是异步向量，它也是转变成同步向量而采用批处理方式。如果用adapt 函数，输入格式决定着网络训练方式。如果传递的是序列，网络用增加方式，如果传递的是同步向量就采用批处理方式。

在动态网络中，特别是当仅有一个训练序列存在时，批处理方式只能用train函数完成。参照增加方式静态网络中使用的网络，把学习速率设为0.02。

```
net = newlin([-1 1],1,[0 1],0.02);
net.IW{1,1}=[0 0];
net.biasConnect=0;
net.trainParam.epochs = 1;
Pi ={1};
P ={2 3 4};
T ={3 5 6};
```

用以前增加方式训练过的那组数据训练，但是需要只有在所有数据都提交后才更新权重（批处理方式）。因为输入是一个序列，网络将用异步模式模拟，但是权重将用批处理方式更新。

```
net=train(net,P,T,Pi);
```

经过一次训练后，权重值如下。

>>net.IW{1,1}

ans = 0.9000 0.6200

这里的权重值和我们用增加方式得到的不同。在增加方式中，通过训练设置，一次训练可以更新权重3次。在批处理方式中，每次训练只能更新1次。

7.1.4 BP神经网络功能

目前，在人工神经网络的实际应用中。绝大部分的神经网络模型都采用 BP 神经网络及其变化形式。它也是前向网络的核心部分，体现了人工神经网络的精华。

BP网络主要用于以下4方面。

·函数逼近：用输入向量和相应的输出向量训练一个网络以逼近一个函数。

·模式识别：用一个待定的输出向量将它与输入向量联系起来。

·分类：把输入向量所定义的合适方式进行分类。

·数据压缩：减少输出向量维数以便传输或存储。

7.2 网络的设计

7.2.1 网络的层数

在理论上，具有偏差和至少一个S型隐含层加上一个线性输出层的网络，能够逼近任何有理数。增加层数可以提高精度降低误差，但同时也使网络复杂化，从而增加了网络权值的训练时间。

误差精度的提高也可以通过增加神经元数目来获得，其训练效果比增加层数更容易观察和调整。在一般情况下，应优先考虑增加隐含层中的神经元数。

7.2.2 隐含层的神经元数

增加神经元数或者隐含层数，可以提高网络训练的精度。但是在结构实现上，增加神经元数要比增加隐含层数简单。

究竟选取多少隐含层节点合适，目前在理论上还没有一个明确的规定。在具体设计时，一般采取尝试法，即通过对不同神经元数进行训练对比，然后选取最适合的网络。

7.2.3 初始权值的选取

由于系统是非线性的，初始值会在很大程度上影响学习是否达到局部最小、是否能够收敛及训练时间的长短。

如果初始值太大，使得加权后的输入和n落在了S型激活函数的饱和区，从而导致其导数 $f'(n)$非常小，而在计算权值修正公式 $'\ (\ )f\ \ n\delta\propto$ 中，因为，当 $f'(n)\rightarrow 0$时，则有$\delta\rightarrow 0$。

在一般情况下，希望经过初始加权后的每个神经元的输出值都接近于零，这样可以保证每个神经元的权值都能够在它们的S型激活函数变化最大之处进行调节。所以，一般取初始权值在[-1 1]之间的随机数。

7.2.4 学习速率

学习速率决定每一次循环训练中所产生的权值变化量。大的学习速率可能导致系统的不稳定；但小的学习速率导致较长的训练时间，可能收敛很慢，不过能保证网络的误差值不跳出误差表面的低谷而最终趋于最小误差值。

在一般情况下，倾向于选取较小的学习速率以保证系统的稳定性。学习速率的选取范围在0.01～0.8之间。

7.3 BP神经网络工具箱函数

神经网络工具箱几乎涵盖了所有的神经网络的基本常用模型，如感知器和BP网络等。对于各种不同的网络模型，神经网络工具箱集成了多种学习算法，为用户提供了极大的方便。MATLAB神经网络工具箱中包含了许多用于BP网络分析与设计的函数，BP网络的常用函数见表1。

表7-1 BP网络的常用函数

函数类型	函数名称	函数用途
前向网络创建函数	newcf	创建级联前向网络
	newff	创建前向 BP 网络

续表

函数类型	函数名称	函数用途
传递函数	logsig	S 型的对数函数
	tansig	S 型的正切函数
	purelin	纯线性函数
训练函数	train	神经网络训练函数
	traingd	梯度下降的 BP 算法训练函数
学习函数	learngd	基于梯度下降法的学习函数
	learngdm	梯度下降动量学习函数
性能函数	mse	均方误差函数
	msereg	均方误差规范化函数
显示函数	plotperf	绘制网络的性能
	plotes	绘制一个单独神经元的误差曲面
	plotep	绘制权值和阈值在误差曲面上的位置
	errsurf	计算单个神经元的误差曲面

对于表7-1中出现的函数或命令，下面选择比较重要的几个详细介绍。

1.BP网络创建函数

newff函数用于创建一个BP网络。调用格式如下。

net=newff

net=newff(PR,[S1 S2..SN1],{TF1 TF2..TFN1},BTF,BLF,PF)

net=newff;用于在对话框中创建一个BP网络。

其中，net为创建的新BP神经网络。

PR为网络输入向量取值范围的矩阵。

[S1 S2…SNl]表示网络隐含层和输出层神经元的个数。

{TFl　TF2…TFN1}表示网络隐含层和输出层的传输函数，默认为‘tansig’。

BTF表示网络的训练函数，默认为‘trainlm’。

BLF表示网络的权值学习函数，默认为‘learngdm’。

PF表示性能数，默认为‘mse’。

【例7-1】使用newff函数创建一个BP网络，并评估其性能。

解：建立一个BP网络，并计算其输出误差。MATLAB代码如下。

```
clear all
clc
net=newff([-3 3],[6 1]);
P=[-3 -2 1 0 1 2 3];
T=[0 1 1 0 0 0 1];
net = train(net,P,T);
y=sim(net,P)
e=T-y              %计算误差向量
net.trainParam.epochs= 100;       %设置仿真总步长
perf=msereg(e,net)
```

运行结果如下所示。

y =

0.0000   1.0000   0.5000   -0.0000   0.5000   0.0000   1.0000

e =

-0.0000   0.0000   0.5000   0.0000   -0.5000   -0.0000   0.0000

perf =

1.7529

BP神经网络训练的误差变化曲线如图7-5所示。

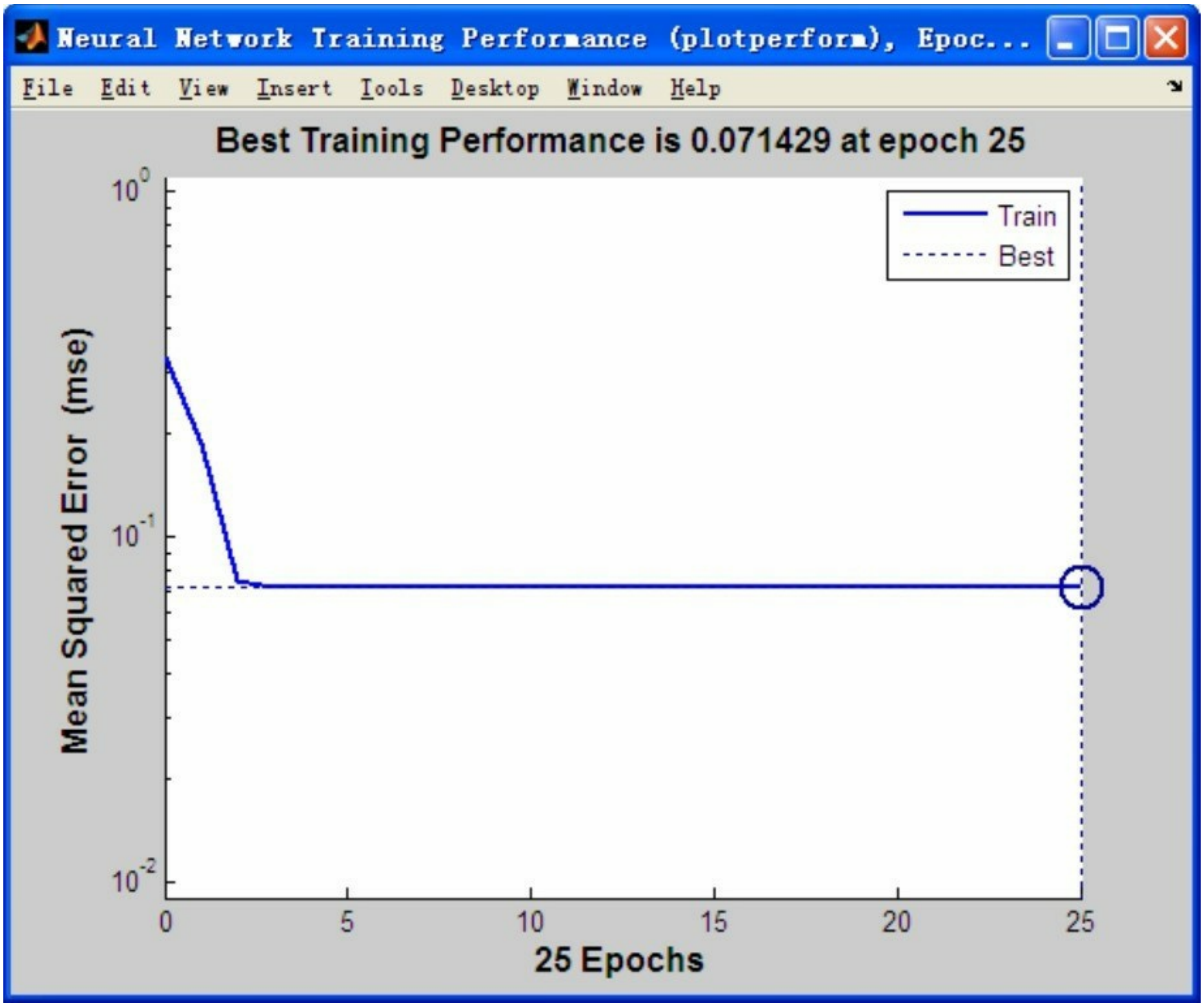


图7-5 BP神经网络训练的误差变化曲线

2.BP网络传递函数

传递函数是BP网络的重要组成部分。传递函数又称为激活函数，必须是连续可微的。BP网络经常采用S型的对数或正切函数和线性函

数。

（1）S型的对数函数logsig。该函数调用格式如下。

A=logsig（N）

info=logsig（code）

其中，N为Q个S维的输入列向量；A为函数返回值，位于区间（0，1）中。

（2）双曲正切S型传递函数tansig。该函数调用格式如下。

A=tansig（N）

info=tansig（code）

其中，N为Q个S维的输入列向量；A为函数返回值，位于区间（−1，1）之间。

（3）线性传递函数purelin。该函数调用格式如下。

A=purelin（N）

info=purelin（code）

其中，N为Q个S维的输入列向量；A为函数返回值，A=N。

3.BP网络学习函数

（1）learngd函数为梯度下降权值/阈值学习函数，它通过神经元的输入和误差，以及权值和阈值的学习效率，来计算权值或阈值的变化率。调用格式如下。

[dW,ls]=learngd(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)

[db,ls]=learngd(b,ones(1,Q),Z,N,A,T,E,gW,gA,D,LP,LS)

info=learngd(code)

（2）learngdm函数为梯度下降动量学习函数，它利用神经元的输入和误差、权值或阈值的学习速率和动量常数，来计算权值或阈值的变化率。

4.BP网络训练函数

（1）train函数为神经网络训练函数，调用其他训练函数，对网络

进行训练。该函数的调用格式如下。

[net,tr,Y,E,Pf,Af]=train(NET,P,T,Pi,Ai)

[net,tr,Y,E,Pf,Af]=train(NET,P,T,Pi,Ai,VV,TV)

（2）traingd函数为梯度下降BP算法函数。traingdm函数为梯度下降动量BP算法函数。

5.BP网络显示函数

（1）plotes函数用于绘制一个单独的神经元误差曲面。其调用格式如下所示。

plotes (WV,BV,ES,V)

其中，WV表示权值的N维向量。

BV表示M维的阈值向量。

ES表示误差向量组成的M×N维矩阵。

V 表示曲面的视角，默认为[-37.5, 30]。

函数绘制的误差曲面是由权值和阈值确定的，并由函数errsurf计算得到。

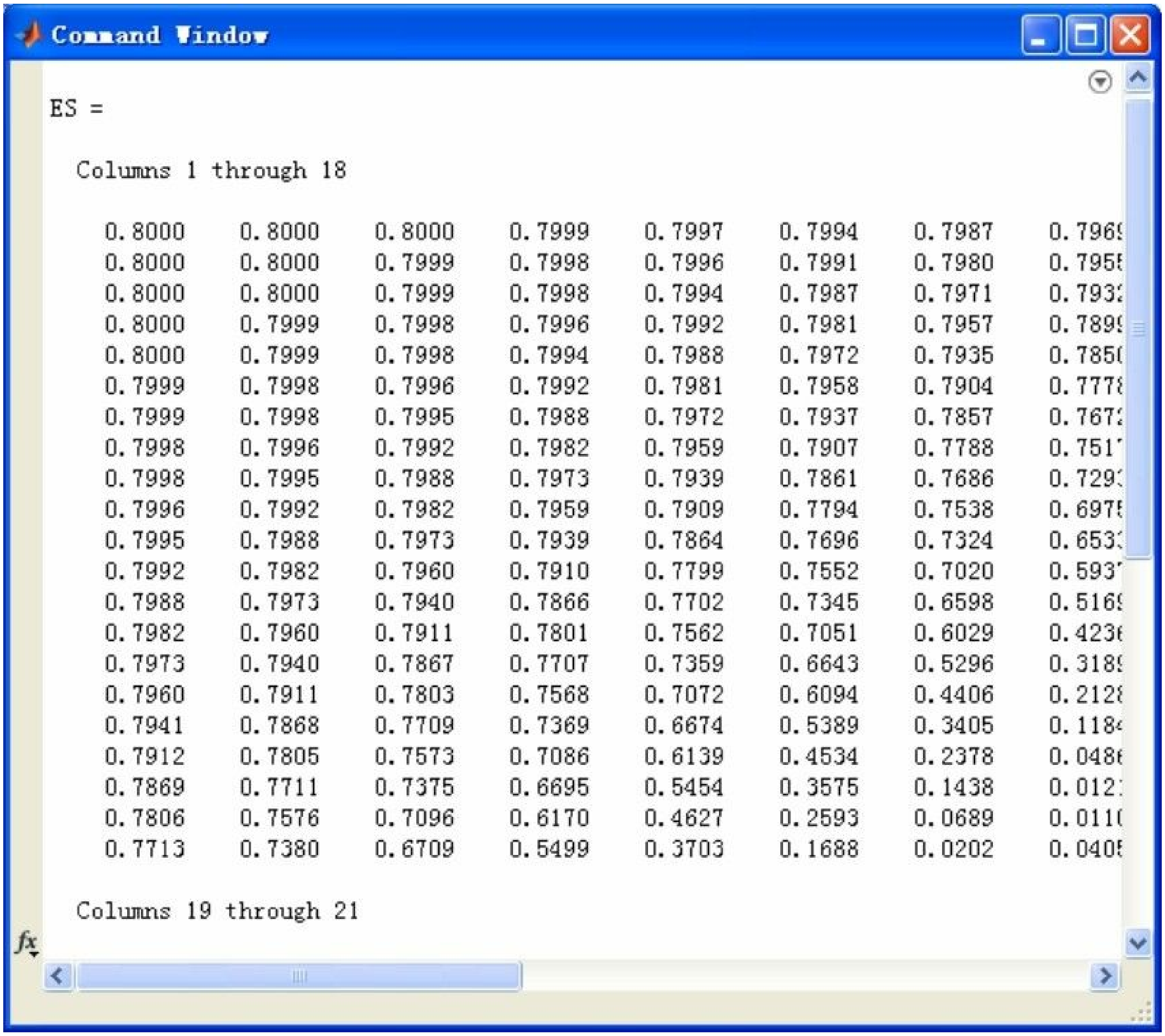
【例7-2】利用函数 plotes 绘制一个神经网络的误差曲面，传递函数选取 logsig 对数函数。

解：MATLAB程序如下所示。

```
clear all
clc
p = [3 2];
t = [0.4 0.8];
wv =-4:0.4:4; bv = wv;
ES = errsurf(p,t,wv,bv,'logsig');
plotes(wv,bv,ES,[60 30])
```

运行程序后得到ES值如图7-6所示。

函数plotes绘制的神经网络误差曲面如图7-7所示。




```
ES =

  Columns 1 through 18

    0.8000    0.8000    0.8000    0.7999    0.7997    0.7994    0.7987    0.796
    0.8000    0.8000    0.7999    0.7998    0.7996    0.7991    0.7980    0.795
    0.8000    0.8000    0.7999    0.7998    0.7994    0.7987    0.7971    0.793
    0.8000    0.7999    0.7998    0.7996    0.7992    0.7981    0.7957    0.789
    0.8000    0.7999    0.7998    0.7994    0.7988    0.7972    0.7935    0.785
    0.7999    0.7998    0.7996    0.7992    0.7981    0.7958    0.7904    0.777
    0.7999    0.7998    0.7995    0.7988    0.7972    0.7937    0.7857    0.767
    0.7998    0.7996    0.7992    0.7982    0.7959    0.7907    0.7788    0.751
    0.7998    0.7995    0.7988    0.7973    0.7939    0.7861    0.7686    0.729
    0.7996    0.7992    0.7982    0.7959    0.7909    0.7794    0.7538    0.697
    0.7995    0.7988    0.7973    0.7939    0.7864    0.7696    0.7324    0.653
    0.7992    0.7982    0.7960    0.7910    0.7799    0.7552    0.7020    0.593
    0.7988    0.7973    0.7940    0.7866    0.7702    0.7345    0.6598    0.516
    0.7982    0.7960    0.7911    0.7801    0.7562    0.7051    0.6029    0.423
    0.7973    0.7940    0.7867    0.7707    0.7359    0.6643    0.5296    0.318
    0.7960    0.7911    0.7803    0.7568    0.7072    0.6094    0.4406    0.212
    0.7941    0.7868    0.7709    0.7369    0.6674    0.5389    0.3405    0.118
    0.7912    0.7805    0.7573    0.7086    0.6139    0.4534    0.2378    0.048
    0.7869    0.7711    0.7375    0.6695    0.5454    0.3575    0.1438    0.012
    0.7806    0.7576    0.7096    0.6170    0.4627    0.2593    0.0689    0.011
    0.7713    0.7380    0.6709    0.5499    0.3703    0.1688    0.0202    0.040

  Columns 19 through 21
```

图7-6 神经网络运行得到的ES值

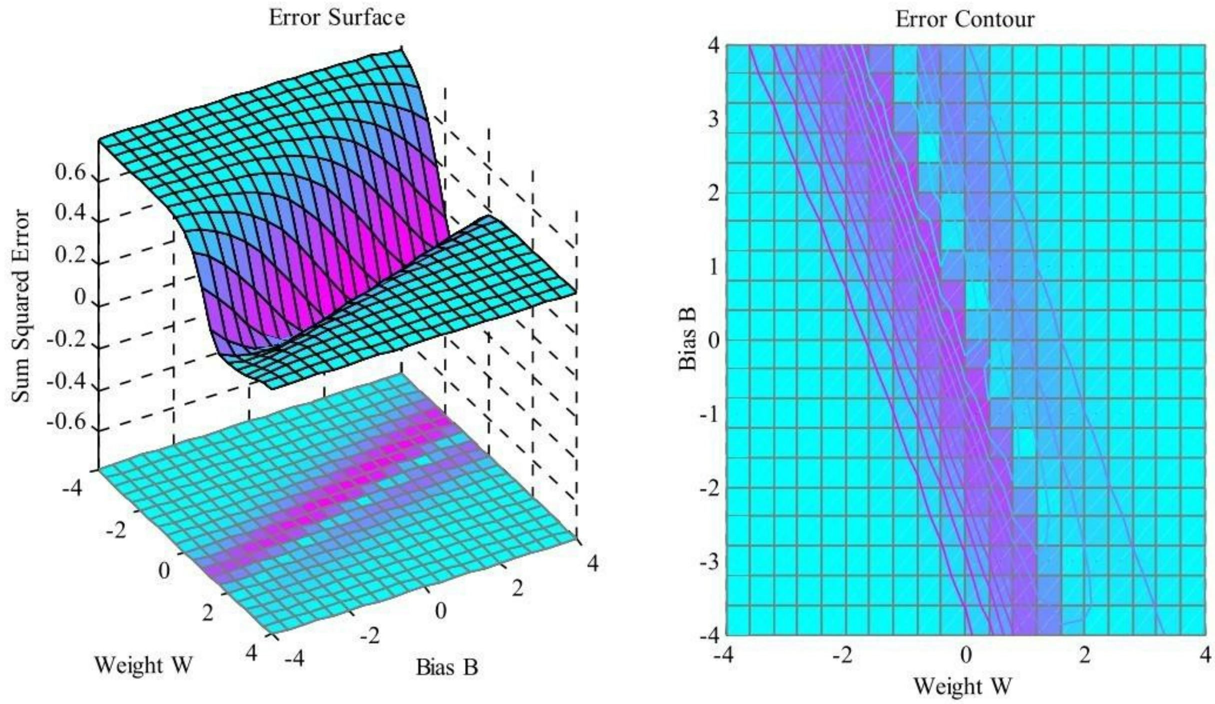


图7-7 神经网络误差曲面

（2）errsurf函数用于计算单个神经元的误差曲面。其调用格式如下。

E=errsurf(P,T,WV,BV,F)

其中，P为输入向量。

T为目标向量。

WV为权值向量。

BV为阈值向量。

F为传递函数的名称。

【例7-3】利用函数errsurf计算单个神经元的误差曲面，要求传递函数选取tansig正切函数。

解：MATLAB程序如下所示。

```
clear all
clc
p = [-6.0 -6.1 -4.1 -4.0 +4.0 +4.1 +6.0 +6.1];
```

```
t = [+0.0 +0.0 +.97 +.99 +.01 +.03 +1.0 +1.0];
wv =-1:.1:1; bv =-2.5:.25:2.5;
es = errsurf(p,t,wv,bv,'tansig');
plotes(wv,bv,es,[60 30])
```

运行程序得到神经网络误差曲面如图7-8所示。

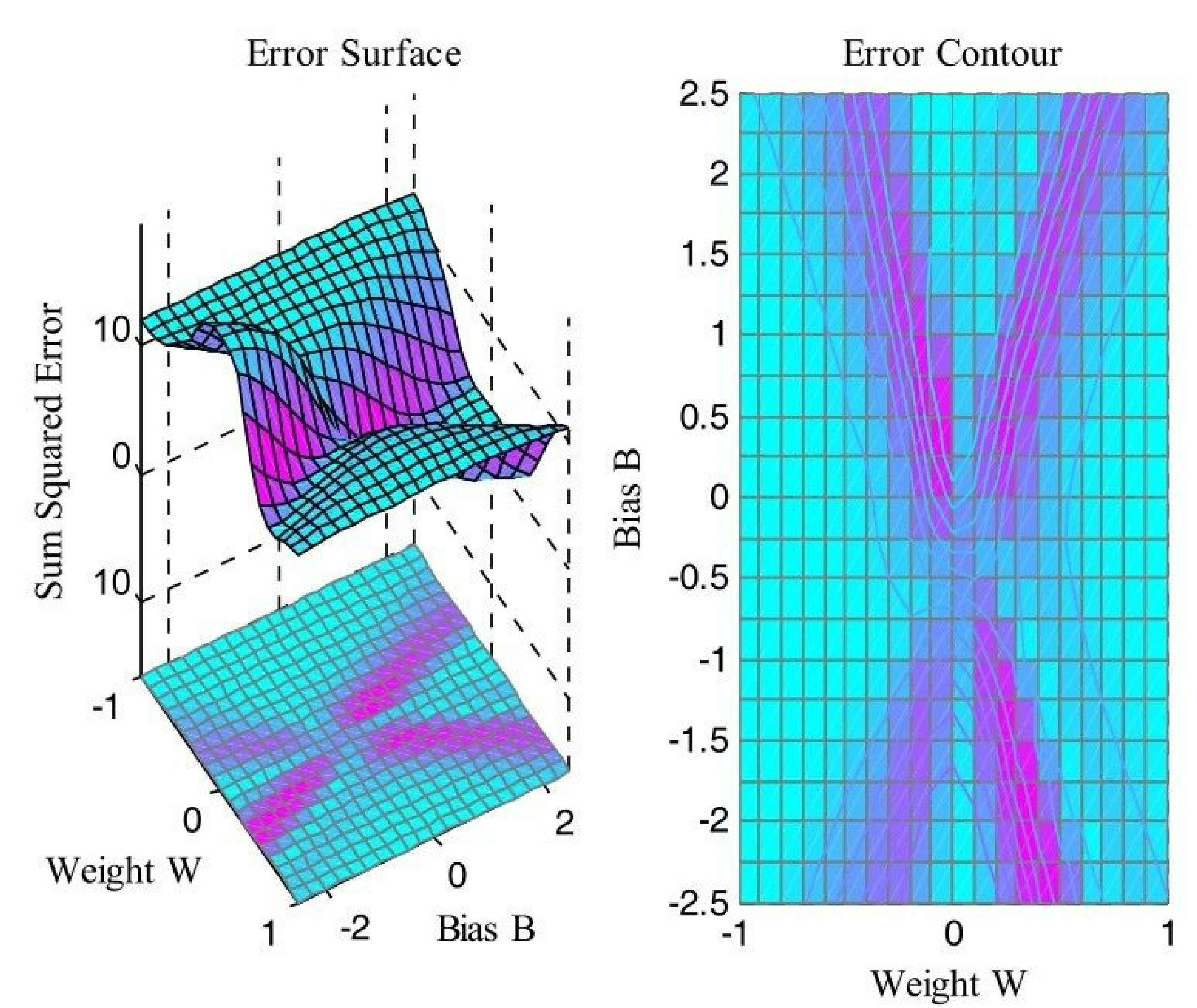


图7-8 神经网络误差曲面

7.4 BP神经网络的工程应用

7.4.1 BP 网络在函数逼近中的应用

BP网络有很强的映射能力，主要用于模式识别分类、函数逼近、函数压缩等。下面将通过实例来说明BP网络在函数逼近方面的应用。

【例7-4】要求设计一个BP网络，逼近以下函数：

$g(x)=1+\sin(k\cdot pi/2\cdot x)$，实现对该非线性函数的逼近。其中，分别令k=2，3，6进行仿真，通过调节参数得出信号的频率与隐含层节点之间、隐含层节点与函数逼近能力之间的关系。

解：假设频率参数k=2，绘制要逼近的非线性函数的目标曲线。MATLAB代码如下。

```
k=2;
p=[-1:.05:8];
t=1+sin(k*pi/2*p);
plot(p,t,'-');
title('要逼近的非线性函数');
xlabel('时间');
ylabel('非线性函数');
```

运行代码后，得到目标曲线如图7-9所示。

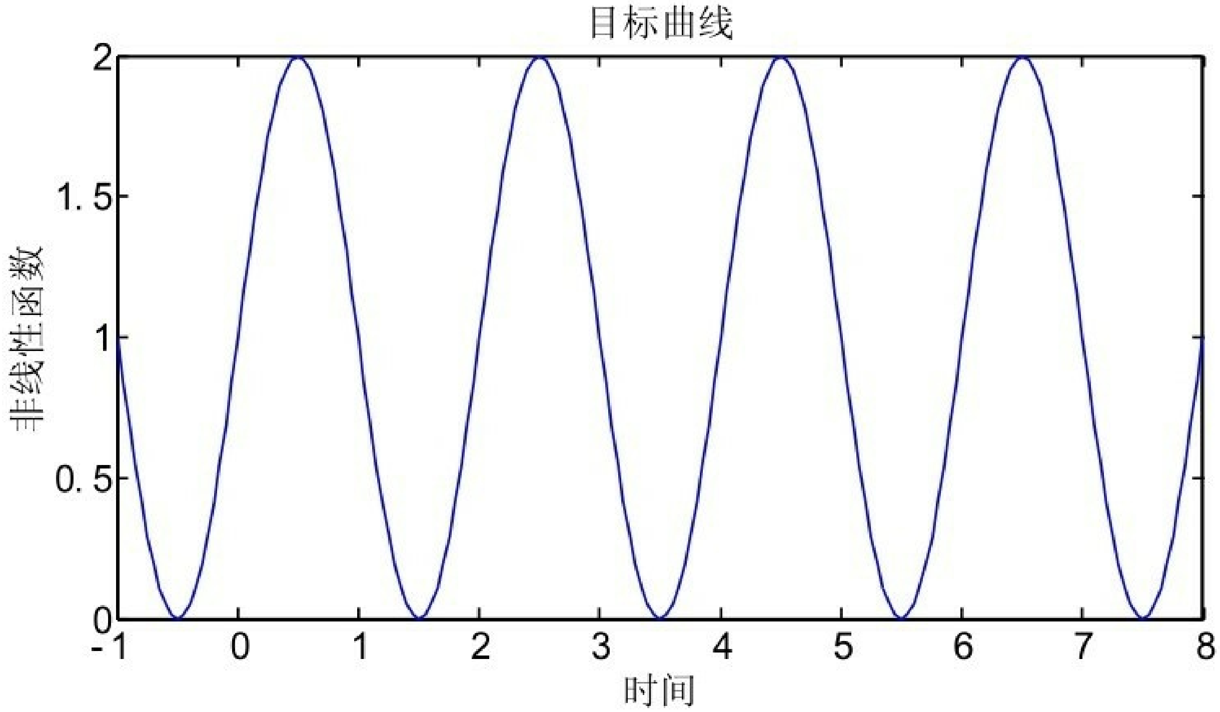


图7-9 逼近的非线性函数曲线

用newff()函数建立BP网络结构。隐含层神经元数目n可以改变，暂设为n=5，输出层有一个神经元。选择隐含层和输出层神经元传递函数分别为tansig函数和purelin函数，网络训练的算法采用Levenberg–Marquardt算法trainlm。

```
n=5;
net = newff(minmax(p),[n,1],{'tansig' 'purelin'},'trainlm');
%对于初始网络，可以应用sim（）函数观察网络输出。
y1=sim(net,p);
figure;
plot(p,t,'-',p,y1,':')
title('未训练网络的输出结果');
xlabel('时间');
ylabel('仿真输出--原函数-');
```

运行上述代码得到网络输出曲线与原函数的比较图，如图7-10所

示。

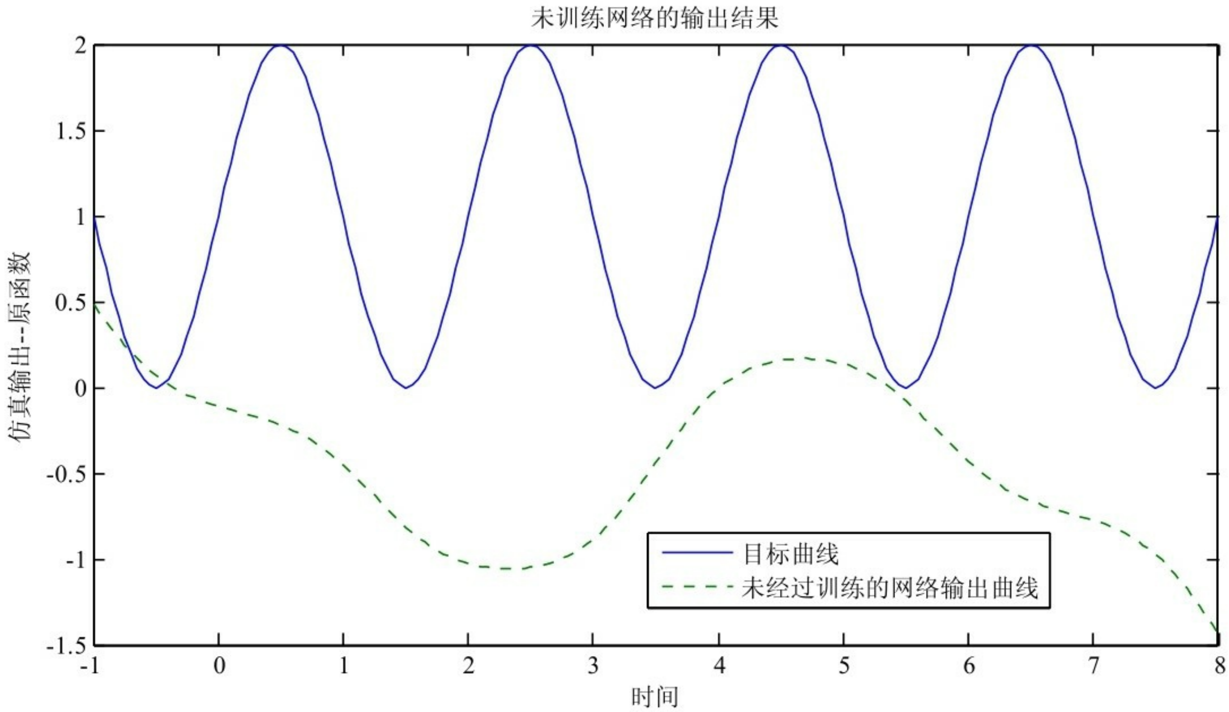


图7-10 网络输出曲线与原函数的比较图

因为使用newff()函数建立函数网络时，权值和阈值的初始化是随机的，所以网络输出结构很差，根本达不到函数逼近的目的，每次运行的结果也有时不同。

应用　　train()函数对网络进行训练之前，需要预先设置网络训练参数。将训练时间设置为200，训练精度设置为0.2，其余参数使用缺省值。训练神经网络的MATLAB代码如下所示。

```
net.trainParam.epochs=200;    %网络训练时间设置为200
net.trainParam.goal=0.2;    %网络训练精度设置为0.2
net=train(net,p,t);        %开始训练网络;
```

训练后得到的误差变化过程如图7-11所示。从图中可以看出，神经网络运行18步后，网络输出误差达到设定的训练精度。

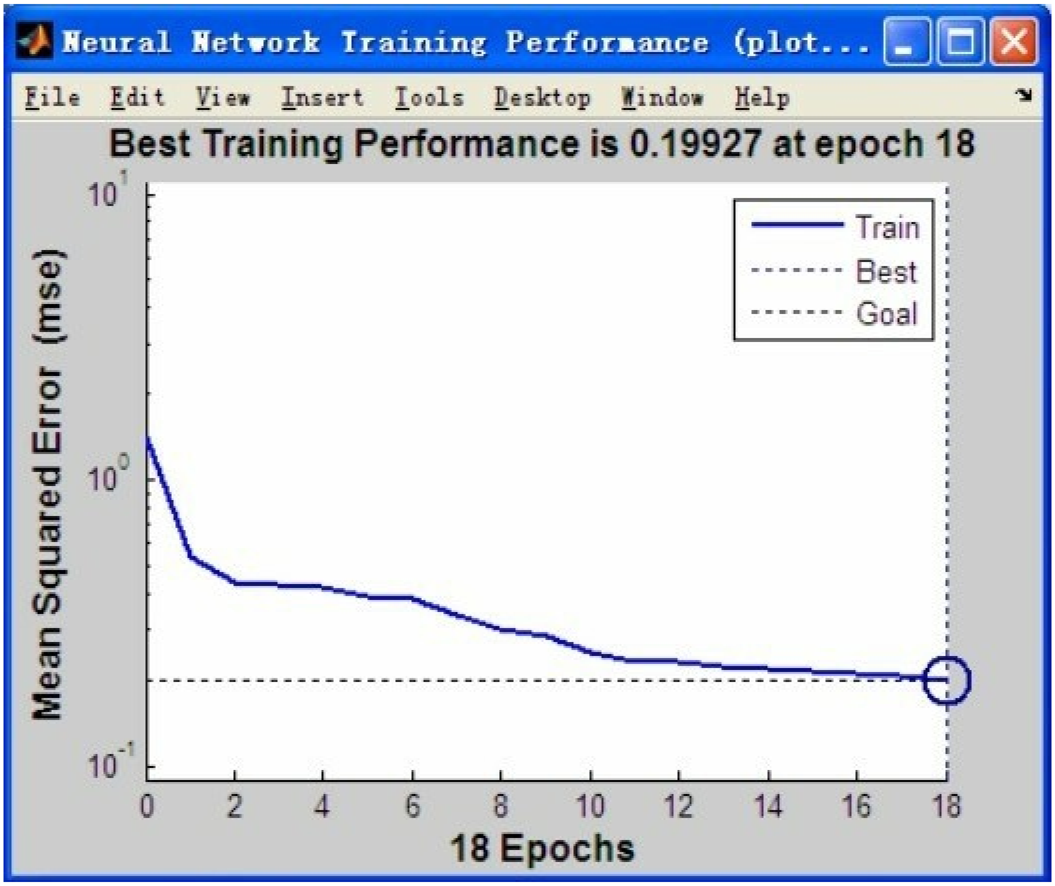


图7-11 训练后得到的误差变化过程

对于训练好的网络进行仿真。

```
y2=sim(net,p);
figure;
plot(p,t,'-',p,y1,':',p,y2, '--')
title('训练后网络的输出结果');
xlabel('时间');
ylabel('仿真输出');
```

绘制网络输出曲线，并与原始非线性函数曲线以及未训练网络的输出结果曲线相比较，比较出来的结果如图7-12所示。

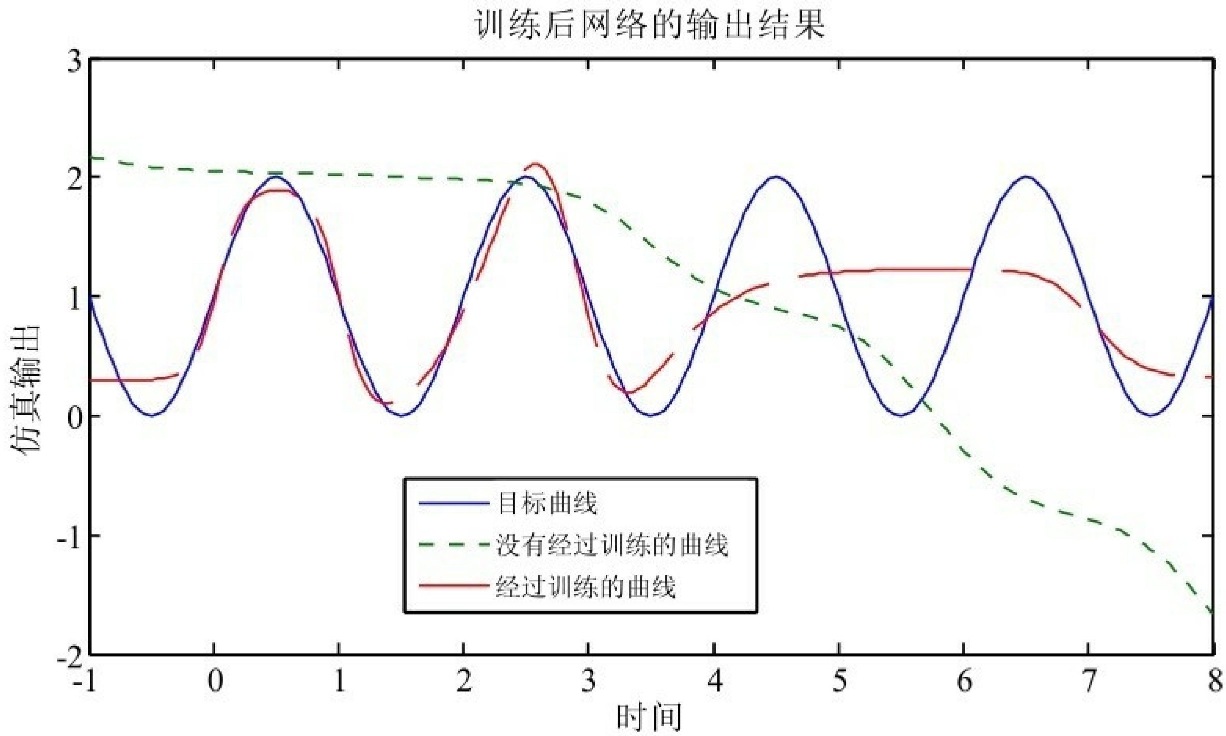


图7-12 训练后网络的输出结果

从图7-12中可以看出，相对于没有训练的曲线，经过训练之后的曲线和原始的目标曲线更接近。这说明经过训练后，BP网络对非线性函数的逼近效果比较好。

改变非线性函数的频率和 BP 函数隐含层神经元的数目，对于函数逼近的效果有一定的影响。

网络非线性程度越高，对于 BP 网络的要求越高，则相同的网络逼近效果要差一些；隐含层神经元的数目对于网络逼近效果也有一定影响，一般来说隐含层神经元数目越多，则BP网络逼近非线性函数的能力越强。

下面通过改变频率参数和非线性函数的隐含层神经元数目来加以比较证明。

（1）频率参数设为k=2，当隐含层神经元数目分别取n=3、n=10时，得到了训练后的网络输出结果，如图7-13和图7-14所示。

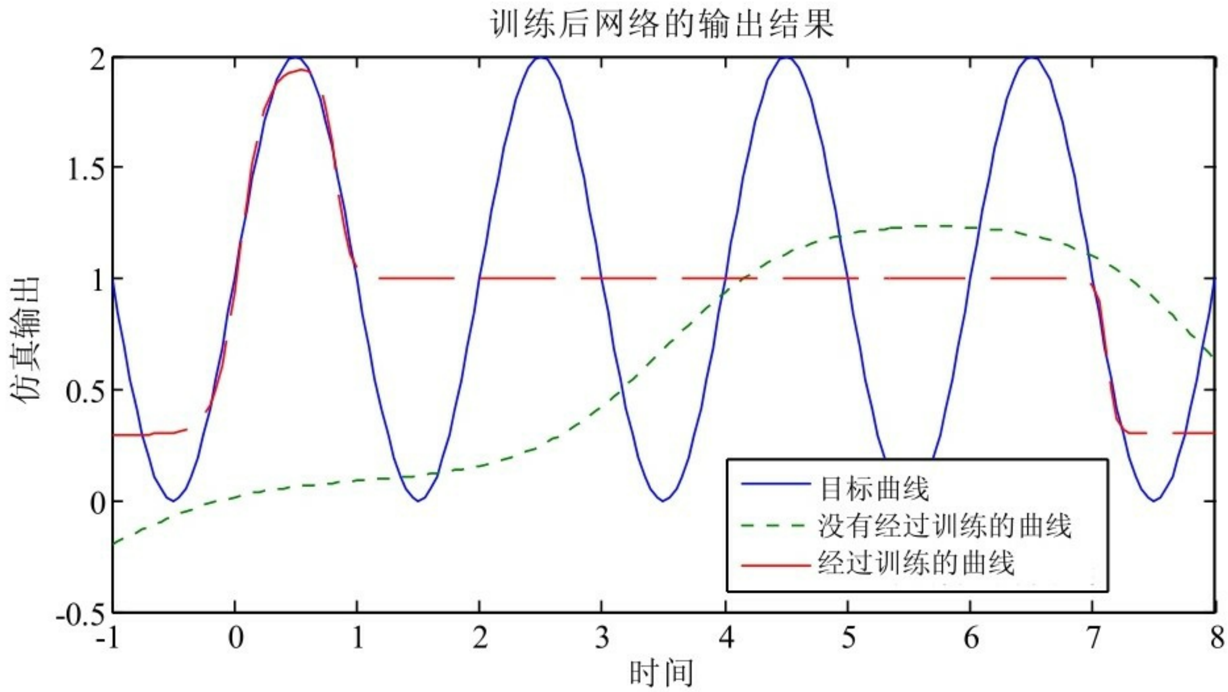


图7-13 当n=3时训练后网络的输出结果

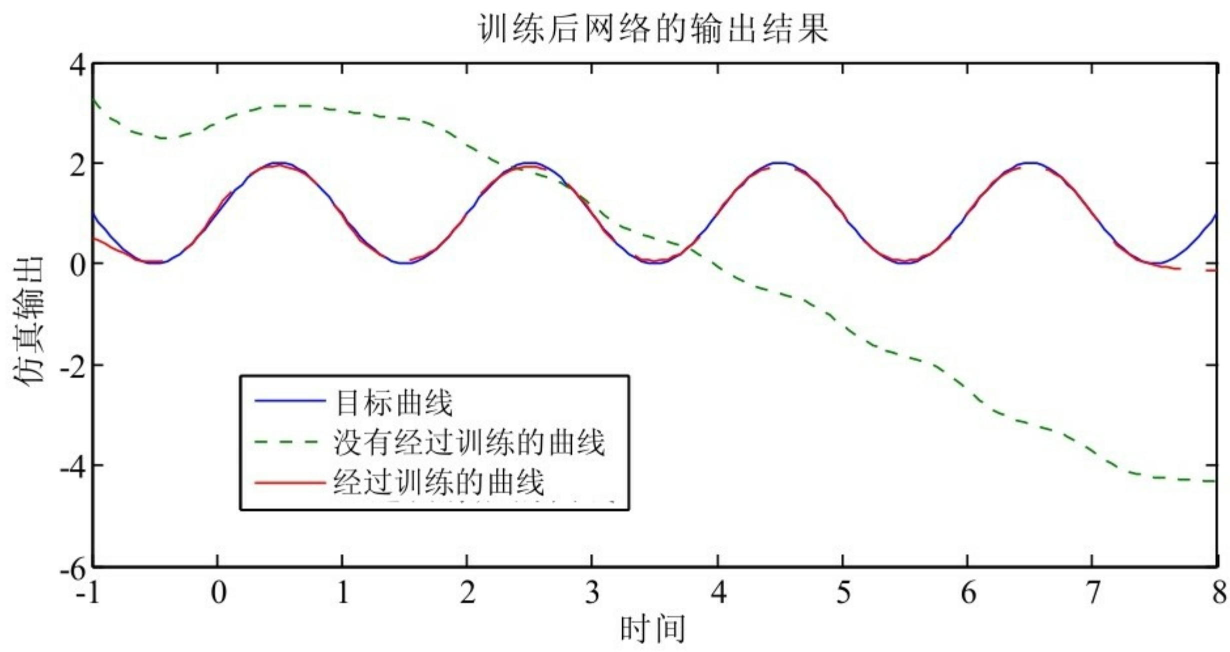


图7-14 当n=10时训练后网络的输出结果

从图中可以看出，当n=10时，经过训练后的曲线基本跟目标曲线重合；当n=3时，经过训练后的曲线基本不跟目标曲线重合。这说明增加隐含层的神经元个数可以增加 BP 神经网络预测的准确性。

（2）频率参数设为k=3，当隐含层神经元数目分别取n=3、n=10时，得到了训练后的网络输出结果，如图7-15和图7-16所示。

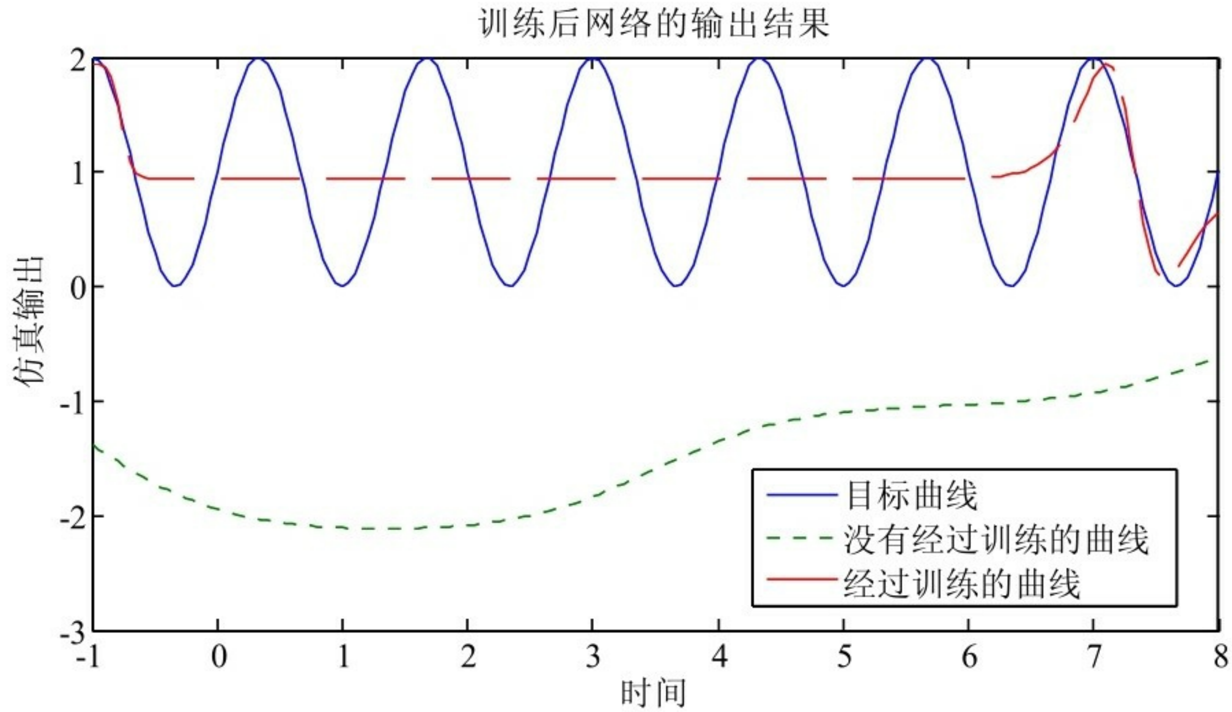


图7-15 当n=3时训练后网络的输出结果

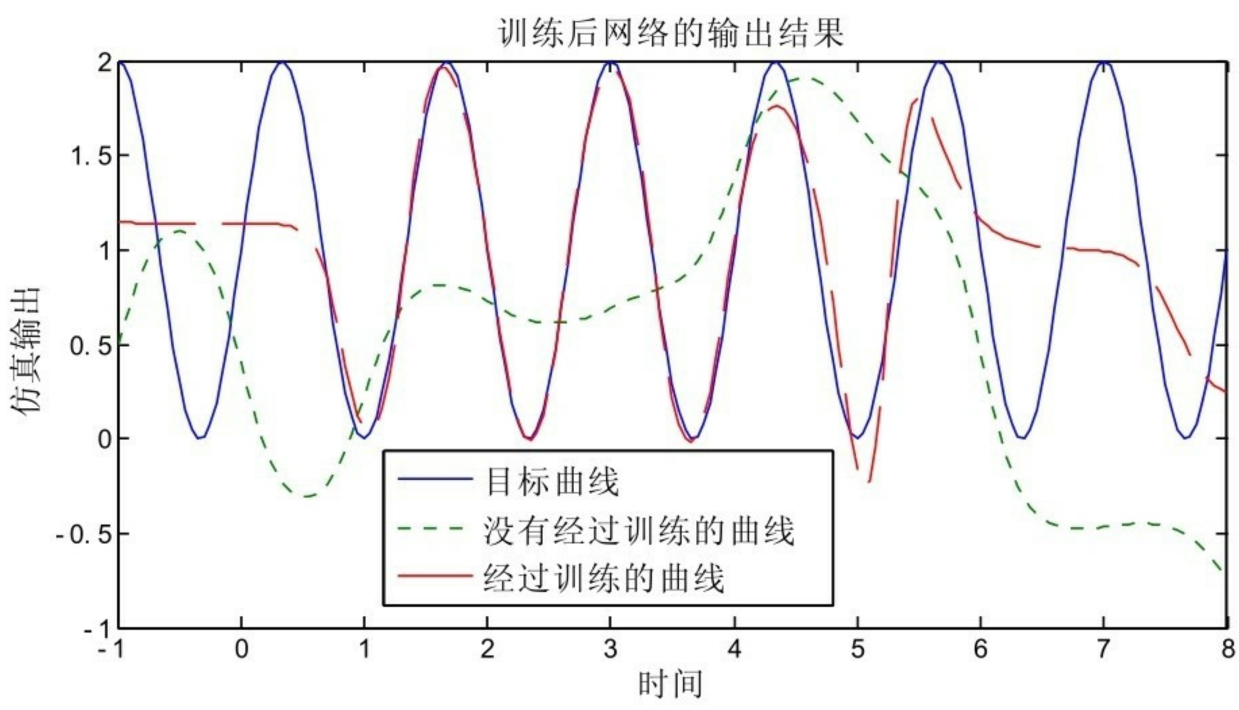


图7-16 当n=10时训练后网络的输出结果

（3）频率参数设为k=6，当隐含层神经元数目分别取n=3、n=10时，得到了训练后的网络输出结果，如图7-17和图7-18所示。

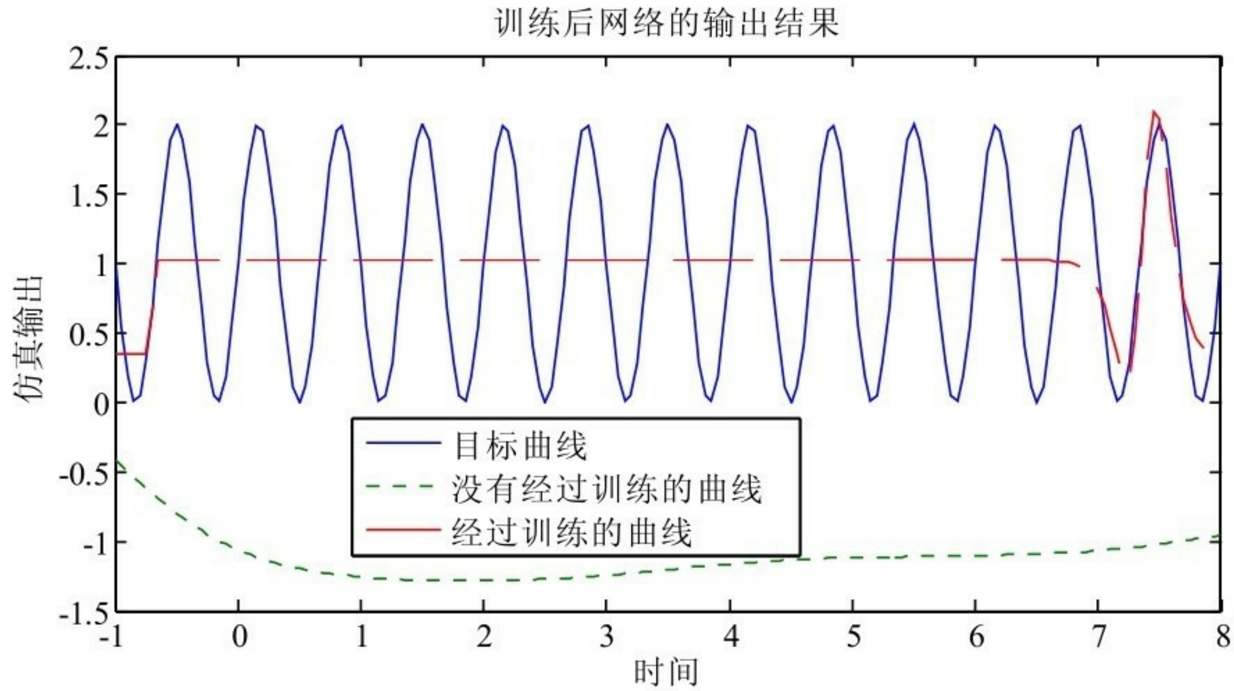


图7-17 当n=3时训练后网络的输出结果

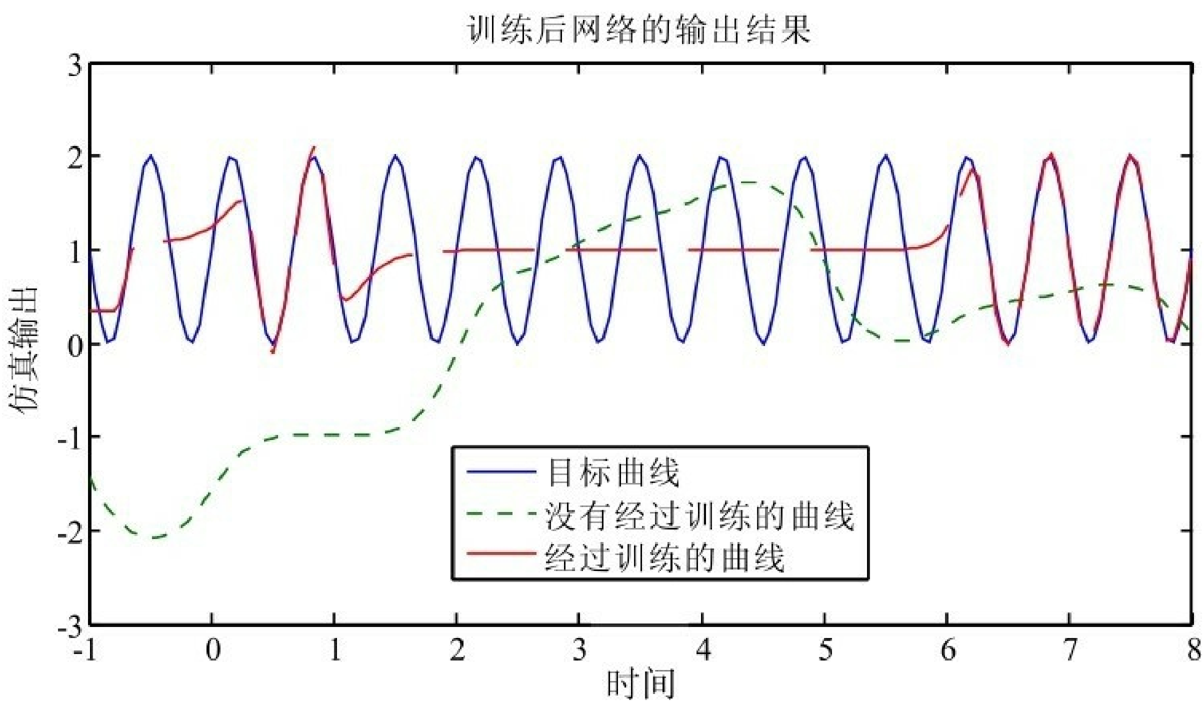


图7-18 当n=10时训练后网络的输出结果

通过上述仿真结果可知，当k=2，n=10时；k=3，n=10时；k=6，n=10时，BP神经网络分别对函数取得了较好的逼近效果。

由此可见，n取不同的值对函数逼近的效果有很大的影响。改变BP网络隐含层神经元的数目，可以改变BP神经网络对于函数的逼近效果。隐含层神经元数目越多，则BP网络逼近非线性函数的能力越强。

由于单隐含层的 BP 神经网络可以逼近任意的非线性映射，在隐含层的神经元个数可以随意调整的前提下。输入层和输出层神经元个数为1，只有一个隐含层，其个数根据上述的设计经验公式和本例的实际情况，选取9～16之间。

下面的隐含层神经元个数可变的 BP 神经网络，通过误差和训练步数对比确定隐含层个数，并检验隐含层神经元个数对网络性能的影响。

MATLAB程序如下。

```
%选取输入变量x取值范围
x=-4:0.01:4;
%输入目标函数
y1=sin((1/2)*pi*x)+sin(pi*x);
%隐含层的神经元数目范围
s=9:16;
%欧氏距离
res=1:8;
%选取不同的隐含层神经元个数，进行网络测试
for i=1:8
%建立前向型BP神经网络，输入层和隐含层激励函数为tansig，输出层为purelin
%训练函数为trainlm，也是默认函数
net=newff(minmax(x),[1,s(i),1],{'tansig','tansig','purelin'},'trainlm');
%训练步数最大为2000
```

net.trainparam.epochs=2000;

%设定目标误差为0.00001

net.trainparam.goal=0.00001;

%进行函数训练

net=train(net,x，y1);

%对训练后的神经网络进行仿真

y2=sim(net,x);

%求欧式距离，判定隐含层神经元个数及网络性能

err=y2-y1;

res(i)=norm(err);

end

根据BP神经网络的MATLAB设计，可以得出下面的通用的MATLAB程序段，由于各种BP学习算法采用了不同的学习函数，所以只需要更改学习函数即可。

MATLAB程序段如下。

x=-4:0.01:4;

y1=sin((1/2)*pi*x)+sin(pi*x);

%trainlm函数可以选择替换

net=newff(minmax(x),[1,15,1],{'tansig','tansig','purelin'},'trainlm');

net.trainparam.epochs=2000;

net.trainparam.goal=0.00001;

net=train(net,x,y1);

y2=sim(net,x);

err=y2-y1;

res=norm(err);

%绘图，原图（蓝色光滑线）和仿真效果图（红色+号点线）

plot(x,y1);

hold on

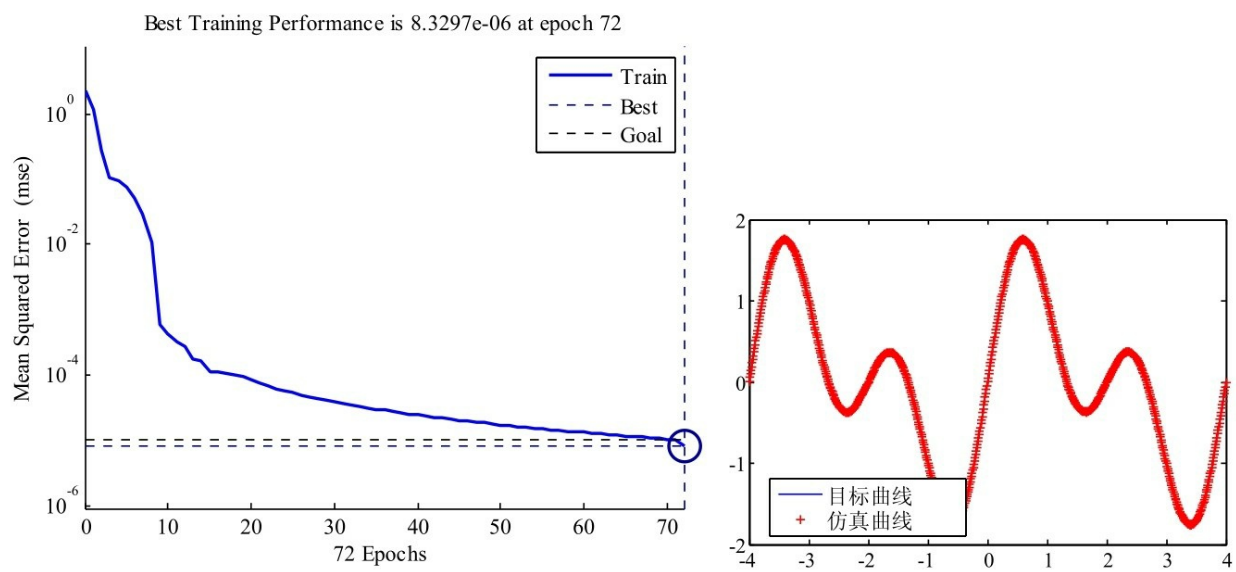
plot(x,y2,'r+');

注意：由于各种不确定因素，可能对网络训练有不同程度的影响，产生不同的效果。

以下为在程序中更换不同的学习算法之后，得到的误差曲线和训练后网络仿真输出结果。

（1）trainlm算法。

trainlm算法训练仿真得到的网络误差曲线和网络仿真曲线如图7-19所示。

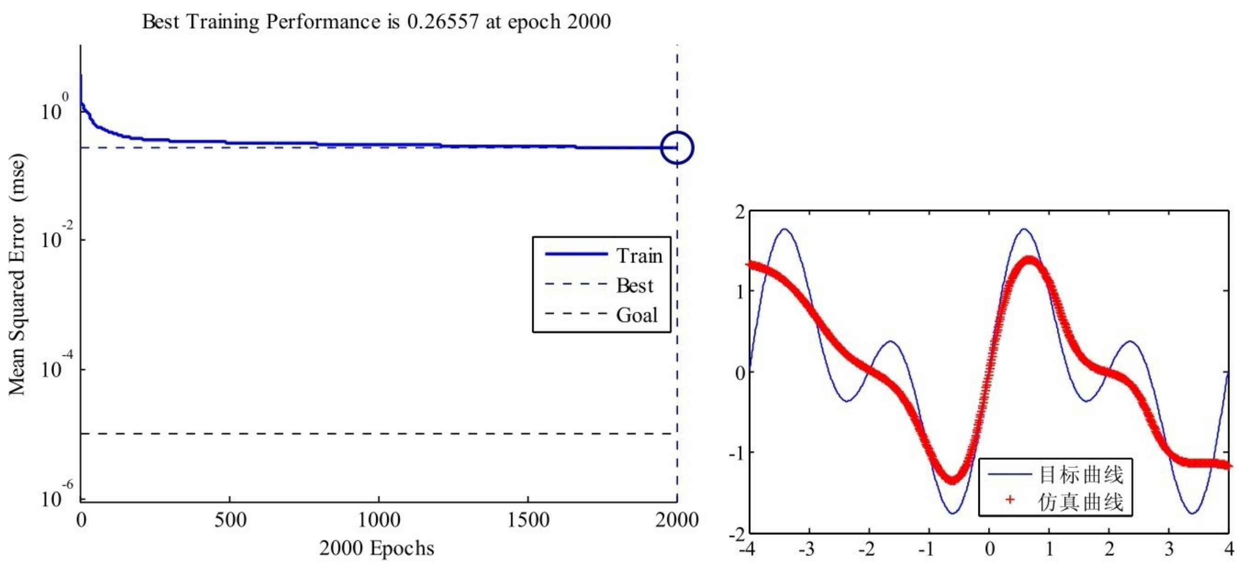


（a） 网络训练误差曲线　　（b） 网络仿真曲线

图7-19 trainlm算法训练仿真所得曲线

（2）traingdm算法。

traingdm算法训练仿真得到的网络误差曲线和网络仿真曲线如图7-20所示。

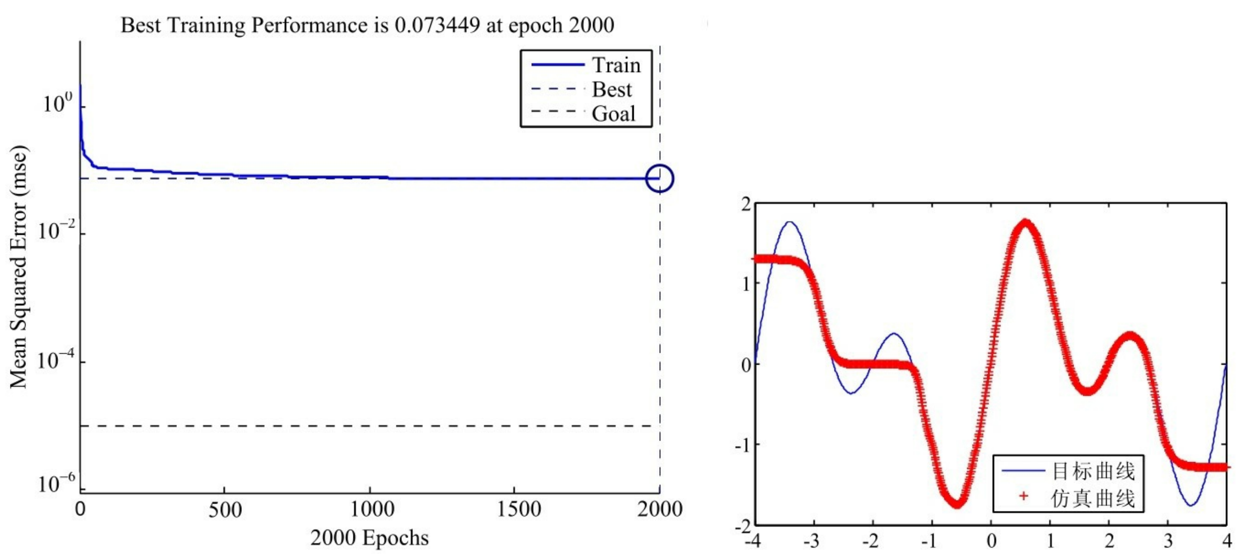


（a） 网络训练误差曲线　　　　（b） 网络仿真曲线

图7-20 traingdm算法训练仿真所得曲线

（3）trainrp算法。

trainrp算法训练仿真得到的网络误差曲线和网络仿真曲线如图7-21所示。



（a） 网络训练误差曲线　　　　（b） 网络仿真曲线

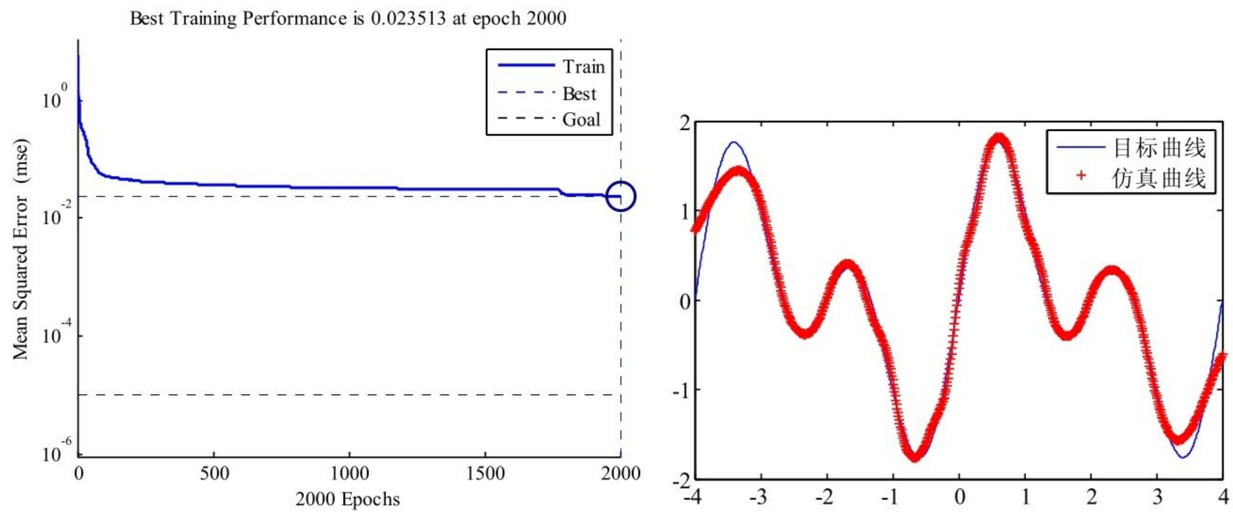
图7-21 trainrp算法训练仿真所得曲线

（4）traingdx算法。

traingdx算法训练仿真得到的网络误差曲线和网络仿真曲线如图7-22所示。

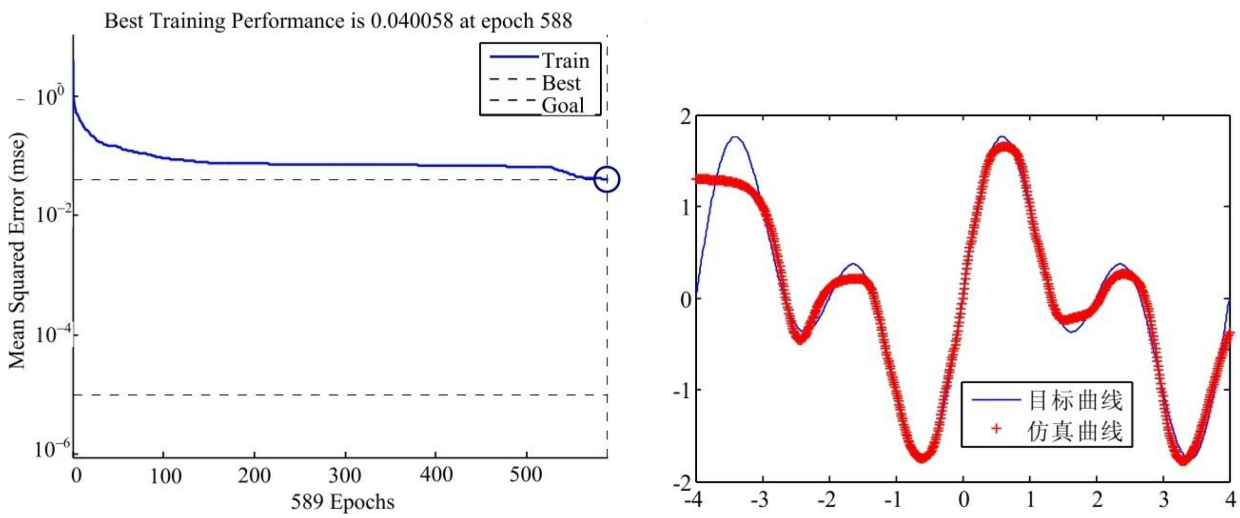
（5）traincgf算法。

traincgf算法训练仿真得到的网络误差曲线和网络仿真曲线如图7-23所示。



（a） 网络训练误差曲线　　　　（b） 网络仿真曲线

图7-22 traingdx算法训练仿真所得曲线



（a） 网络训练误差曲线　　　　（b） 网络仿真曲线

图7-23 traincgf算法训练仿真所得曲线

7.4.2 nntool神经网络工具箱的应用

【例7-5】现给出一药品商店一年当中12个月的药品销售量（单

位：箱），如下。

2356 2095 2400 2398 2634 1600 1873 1487 1900 1500 2046 1556

训练一个BP网络，用当前的所有数据预测下一个月的药品销售量。

解：本题有两种方法实现，一种是编写MATLAB程序，另一种是使用nntool工具箱。

方法一、MATLAB程序实现

我们用前3个月的销售量预测下一个月的销售量，也就是用1～3月的销售量预测第4个月的销售量，用2～4个月的销售量预测第5个月的销售量，如此循环下去，直到用9～11月预测12月份的销售量。这样训练BP神经网络后，就可以用10～12月的数据预测来年1月的销售量。

MATLAB程序如下。

```
p=[2356 2395 2600;2095 2600 2298;2400 2298 1634;2398 1634 1600;2634 1600 1873;1600 1873 1478;1873 1478 1900;1478 1900 1500;1900 1500 2046;]
t=[2298 1634 1600 1873 1487 1900 1500 2046 1556];
pmax=max(p);pmax1=max(pmax);
pmin=min(p);pmin1=min(pmin);
for i=1:9        %归一化处理
   p1(i,:)=(p(i,:)-pmin1)/(pmax1-pmin1);
end
t1=(t-pmin1)/(pmax1-pmin1);
t1=t1';
net=newff([0 1;0 1;0 1],[7 1],{'tansig','logsig'},'traingd');
for i=1:9
net.trainParam.epochs=15000;
net.trainParam.goal=0.01;
```

```
LP.lr=0.1;
net=train(net,p1(i,:)',t1(i));
end
y=sim(net,[1500 2046 1556]');
y1=y*(pmax1-pmin1)+pmin1;
```

如果神经网络的训练函数使用trainlm，则仿真步骤会很少，但需要较大的系统内存。经过预测，来年1月的销售量即上面MATLAB代码中y1为1.9833e+03箱（每次运行后的结果可能不同）。

方法二、nntool神经网络工具箱的使用

（1）在MATLAB命令窗口键入nntool命令打开神经网络工具箱，如图7-24所示。

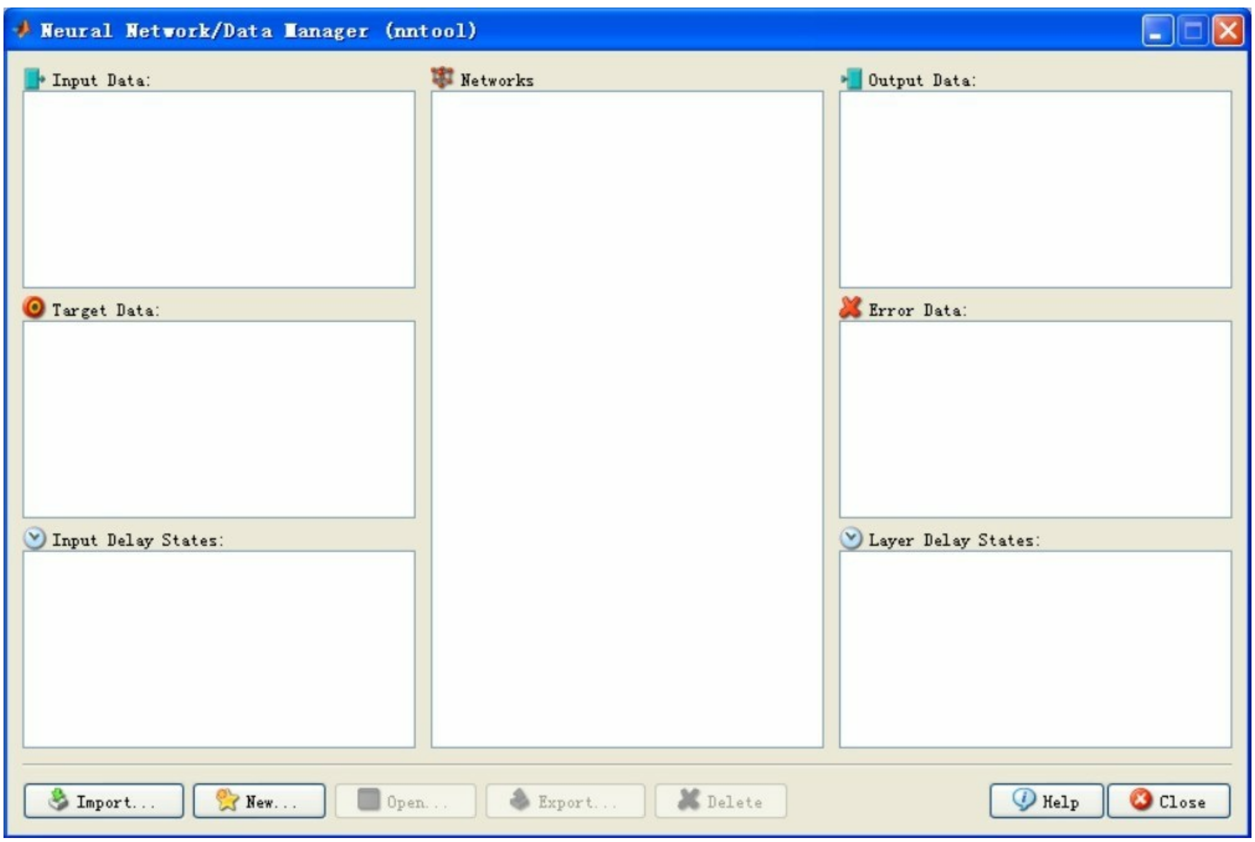


图7-24 神经网络工具箱

（2）单击“Import”按钮，出现如图7-25　所示界面。在Select a Variable选项中，分别把输入向量和目标输出加入到对应的窗口

（[Inputs]和[Targets]）中，有两种可供选择的加入对象（单击 Import 按钮后可以看见），一种是把当前工作区中的某个矩阵加入，另一种是通过.mat文件读入。

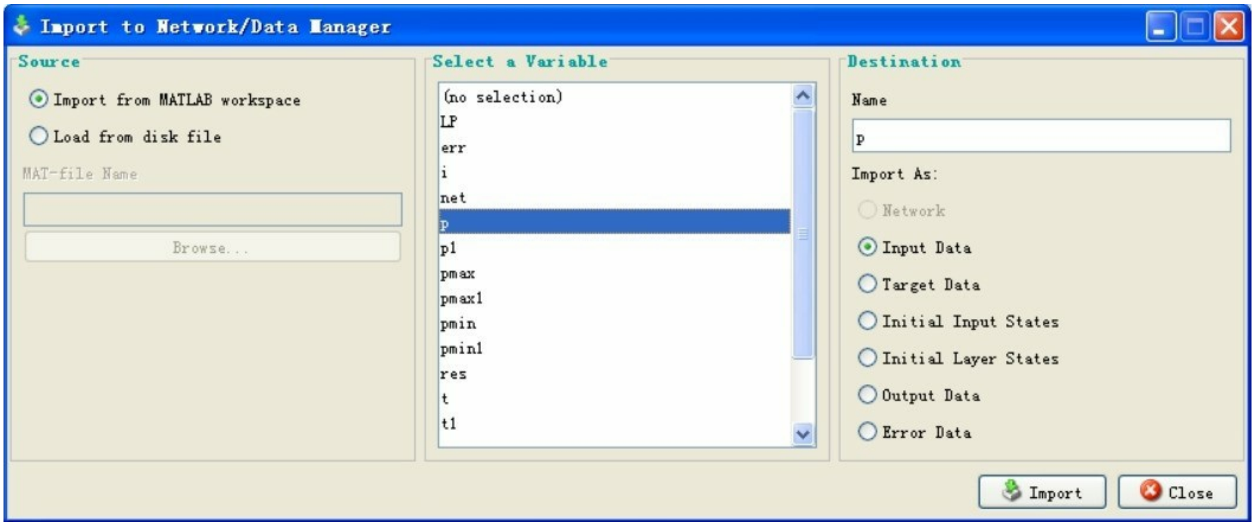


图7-25 输入向量数据导入

（3）单击“New Network”按钮，填入各参数。（以最常用的带一个隐含层的3　层神经网络为例说明，下面没有列出的参数表示使用默认值，例如Network Type 为默认的BP神经网络）。

Input Range——单击Get From Input 下拉框选择加入的输入向量便可自动完成，也可以手动添加。

Training　Function——最好使用TRAINSCG，即共轭梯度法，其好处是当训练不收敛时，它会自动停止训练，而且耗时较其他算法（TRAINLM，TRAINGD）少，也就是收敛很快（如果收敛的话），而且Train Parameters输入不多，也不用太多的技巧调整，一般指定迭代次数、结果显示频率和目标误差即可（详见下文）。

Layer 1 Number of Neurons——隐含层的神经元个数，这是需要经验慢慢尝试并调整的，大致上由输入向量的维数、样本的数量和输出层（Layer2）的神经元个数决定。

一般来说，神经元越多，输出的数值与目标值越接近，但所花费的

训练时间也越长，反之，神经元越少，输出值与目标值相差越大，但训练时间会相应地减少，这是由于神经元越多其算法越复杂造成的，所以需要慢慢尝试，找到一个合适的中间点。

比如输入是3行5000列的0～9的随机整数矩阵，在一开始选择1000个神经元，虽然精度比较高，但是花费的训练时间较长，而且这样神经网络的结构与算法都非常复杂，不容易在实际应用中实现。尝试改为100个，再调整为50个，如果发现在50个以下时精度较差，则可最后定为50个神经元。

Layer 1 Transfer Function——一般用TANSIG（也可以用LOGSIG），即表示隐含层输出是[-1,1]之间的实数，与LOGSIG相比范围更大。

Layer 2 Number of Neurons——输出层的神经元个数，需要与输出的矩阵行数对应，比如设置为3。

Layer 2 Transfer Function——如果是模式识别的两类（或者多类）问题，一般用LOGSIG，即表示输出层的输出是[0,1]之间的实数；如果输出超过[0,1]则可选择PURELIN。

所有参数输入后，可以先用“New…”按钮查看创建的神经网络数据，如图7-26所示。没有问题的话就可以选择“Create”按钮，确认建立神经网络。

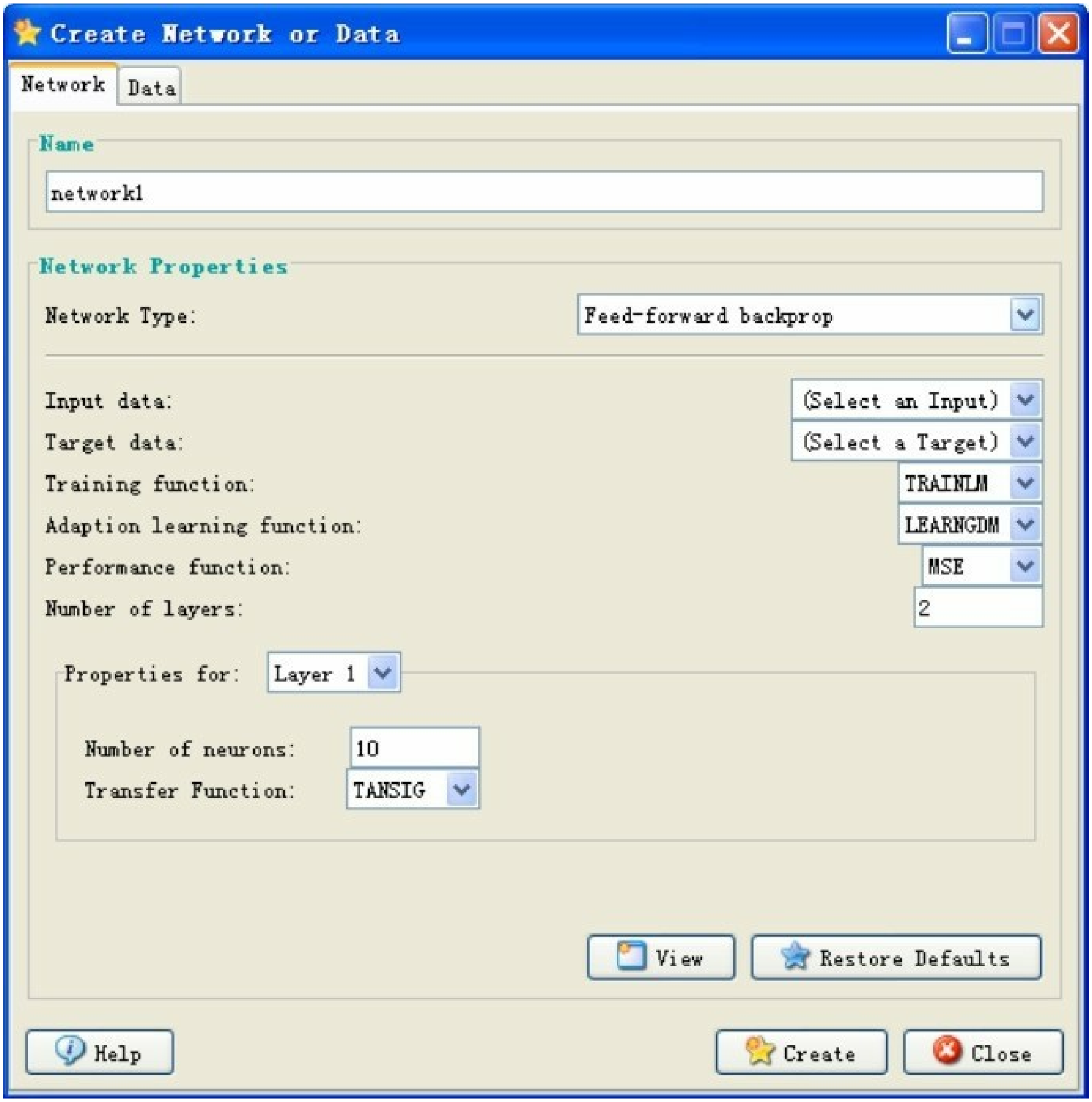


图7-26 创建的神经网络数据

另外，网络创建完毕后，可以单击“View”按钮，查看图7-27所示的网络结构模型。

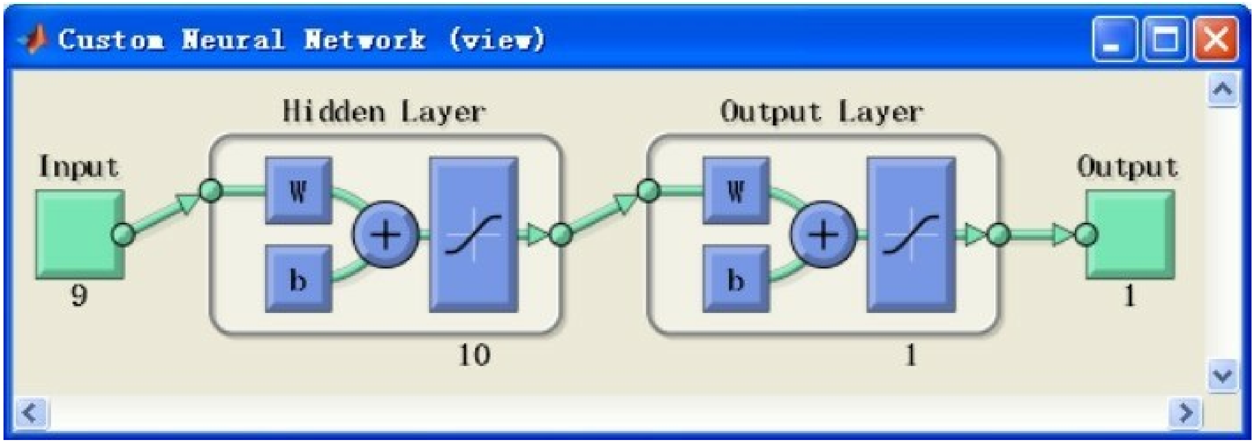


图7-27 建立的网络结构模型

如果需要重新设置权重的初始值，按“Restore Defaults”按钮，这时候MATLAB 执行默认的程序进行权重的初始化。

7.4.3 BP神经网络在语音特征信号分类中的应用

语音特征信号识别是语音识别研究领域中的一个重要方面，一般采用模式匹配的原理解决。语音识别经过语音信号预处理、信号提取、模式匹配和判决规则4步之后，得到识别的结果。

【例7-6】随便选取4种不同的语音信号，用BP神经网络实现对这4种语音信号的有效分类。

解：在本题中，根据倒谱系数法提取4种不同语音的特征信号，不同语音信号分别用1、2、3、4标识，存储于data1.mat、data2.mat、data3.mat和data4.mat的数据库文件中。

在语音信号分类过程中，因为不同语音信号之间有可能存在维数的差别，所以需要进行数据归一化处理。

数据归一化处理是把所有数据转换为[0 1]之间的数，避免因为输入输出数据数量级的差别较大而造成网络预测误差较大。数据归一化处理的方法包括最大最小法和平均数方差法。

MATLAB程序如下所示。

```
clear all
clc
%% 训练数据预测数据提取及归一化
%下载四类语音信号
load data c1
load data c2
load data c3
load data c4
%四个特征信号矩阵合成一个矩阵
data(1:400,:)=c1(1:400,:);
data(401:800,:)=c2(1:400,:);
data(801:1200,:)=c3(1:400,:);
data(1201:1600,:)=c4(1:400,:);
%从1到1600间随机排序
k=rand(1,1600);
[m,n]=sort(k);
%输入输出数据
input=data(:,2:25);
output1 =data(:,1);
%把输出从1维变成4维
for i=1:1600
    switch output1(i)
        case 1
            output(i,:)=[1 0 0 0];
        case 2
            output(i,:)=[0 1 0 0];
        case 3
```

```
            output(i,:)=[0 0 1 0];
        case 4
            output(i,:)=[0 0 0 1];
    end
end
%随机提取1200个样本为训练样本，400个样本为预测样本
input_train=input(n(1:1200),:)';
output_train=output(n(1:1200),:)';
input_test=input(n(1201:1600),:)';
output_test=output(n(1201:1600),:)';
%输入数据归一化
[inputn,inputps]=mapminmax(input_train);
%% 网络结构初始化
innum=24;
midnum=28;
outnum=4;
%权值初始化
w1=rands(midnum,innum);
b1=rands(midnum,1);
w2=rands(midnum,outnum);
b2=rands(outnum,1);
w21=w2;w22=w21;
w11=w1;w12=w11;
b11=b1;b12=b11;
b21=b2;b22=b21;
%学习率
xi=0.2;
```

```
%% 网络训练
for ii=1:10
    E(ii)=0;
    for i=1:1:1200
        %% 网络预测输出
        x=inputn(:,i);
        % 隐含层输出
        for j=1:1:midnum
            I(j)=inputn(:,i)'*w1(j,:)'+b1(j);
            Iout(j)=1/(1+exp(-I(j)));
        end
        % 输出层输出
        yn=w2'*Iout'+b2;
        %% 权值阈值修正
        %计算误差
        e=output_train(:,i)-yn;
        E(ii)=E(ii)+sum(abs(e));
        %计算权值变化率
        dw2=e*Iout;
        db2=e';
        for j=1:1:midnum
            S=1/(1+exp(-I(j)));
            FI(j)=S*(1-S);
        end
        for k=1:1:innum
            for j=1:1:midnum
                dw1(k,j)=FI(j)*x(k)*
```

```
        (e(1)*w2(j,1)+e(2)*w2(j,2)+e(3)*w2(j,3)+e(4)*w2(j,4));
            db1(j)=FI(j)*
        (e(1)*w2(j,1)+e(2)*w2(j,2)+e(3)*w2(j,3)+e(4)*w2(j,4));
          end
        end
        w1=w11+xi*dw1';
        b1=b11+xi*db1';
        w2=w21+xi*dw2';
        b2=b21+xi*db2';
        w1_2=w11;w11=w1;
        w2_2=w21;w21=w2;
        b1_2=b11;b11=b1;
        b2_2=b21;b21=b2;
    end
end
%% 语音特征信号分类
inputn_test=mapminmax('apply',input_test,inputps);
for ii=1:1
    for i=1:400
        %隐含层输出
        for j=1:1:midnum
            I(j)=inputn_test(:,i)'*w1(j,:)'+b1(j);
            Iout(j)=1/(1+exp(-I(j)));
        end
        fore(:,i)=w2'*Iout'+b2;
    end
end
```

```
%% 结果分析
%根据网络输出找出数据属于哪类
for i=1:400
    output_fore(i)=find(fore(:,i)==max(fore(:,i)));
end
%BP网络预测误差
error=output_fore-output1(n(1201:1600))';
%画出预测语音种类和实际语音种类的分类图
figure(1)
plot(output_fore,'r')
hold on
plot(output1(n(1201:1600))','b')
legend('预测语音类别','实际语音类别')
%画出误差图
figure(2)
plot(error)
title('分类误差','fontsize',12)
xlabel('语音信号','fontsize',12)
ylabel('分类误差','fontsize',12)
%print -dtiff -r600 1-4
k=zeros(1,4);
%找出判断错误的分类属于哪一类
for i=1:400
    if error(i)～=0
        [b,c]=max(output_test(:,i));
        switch c
            case 1
```

```
                k(1)=k(1)+1;
            case 2
                k(2)=k(2)+1;
            case 3
                k(3)=k(3)+1;
            case 4
                k(4)=k(4)+1;
        end
    end
end
%找出每类的个体和
kk=zeros(1,4);
for i=1:400
    [b,c]=max(output_test(:,i));
    switch c
        case 1
            kk(1)=kk(1)+1;
        case 2
            kk(2)=kk(2)+1;
        case 3
            kk(3)=kk(3)+1;
        case 4
            kk(4)=kk(4)+1;
    end
end
%正确率
rr=(kk-k)./kk
```

运行以上程序，得到图7-28所示预测语音类别与实际语音类别的比较图。

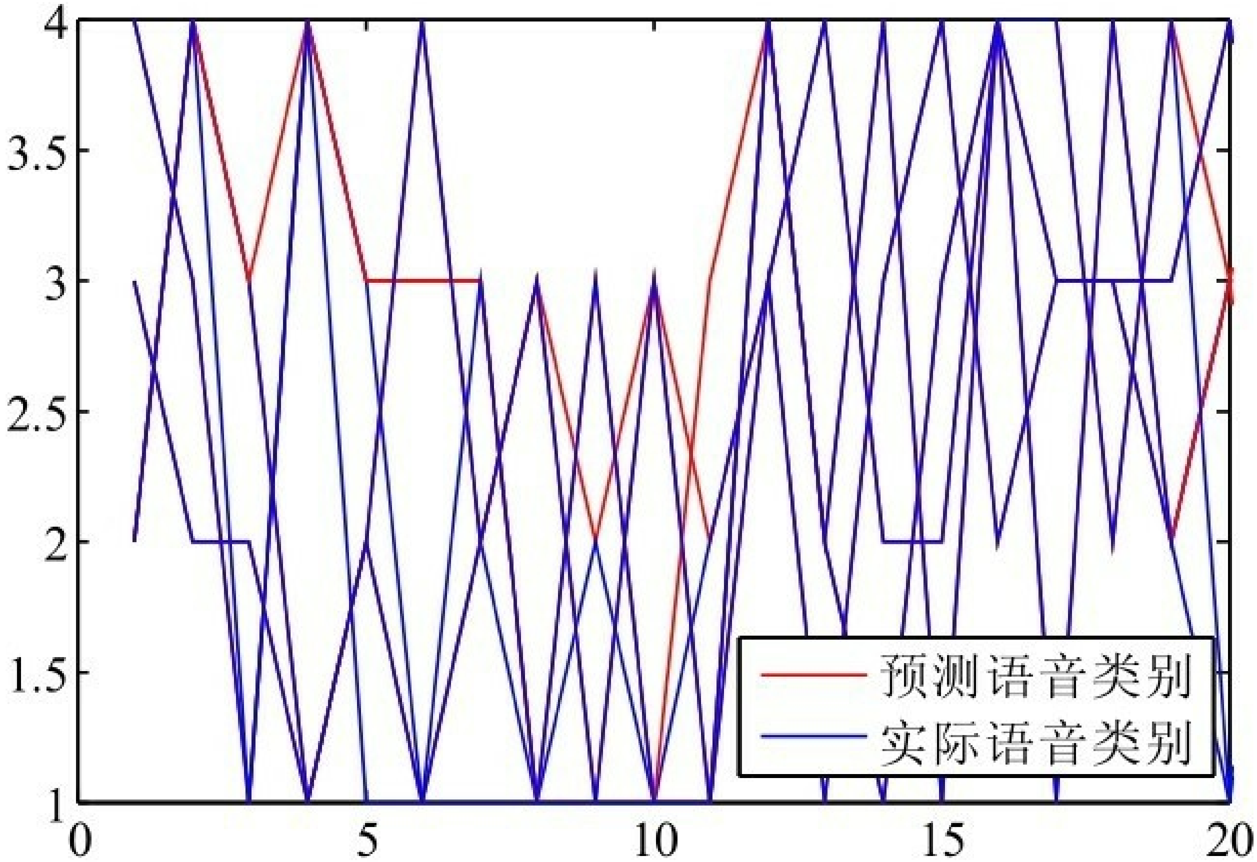


图7-28 预测语音类别与实际语音类别的比较图

仿真得到的语音分类误差曲线如图7-29所示。

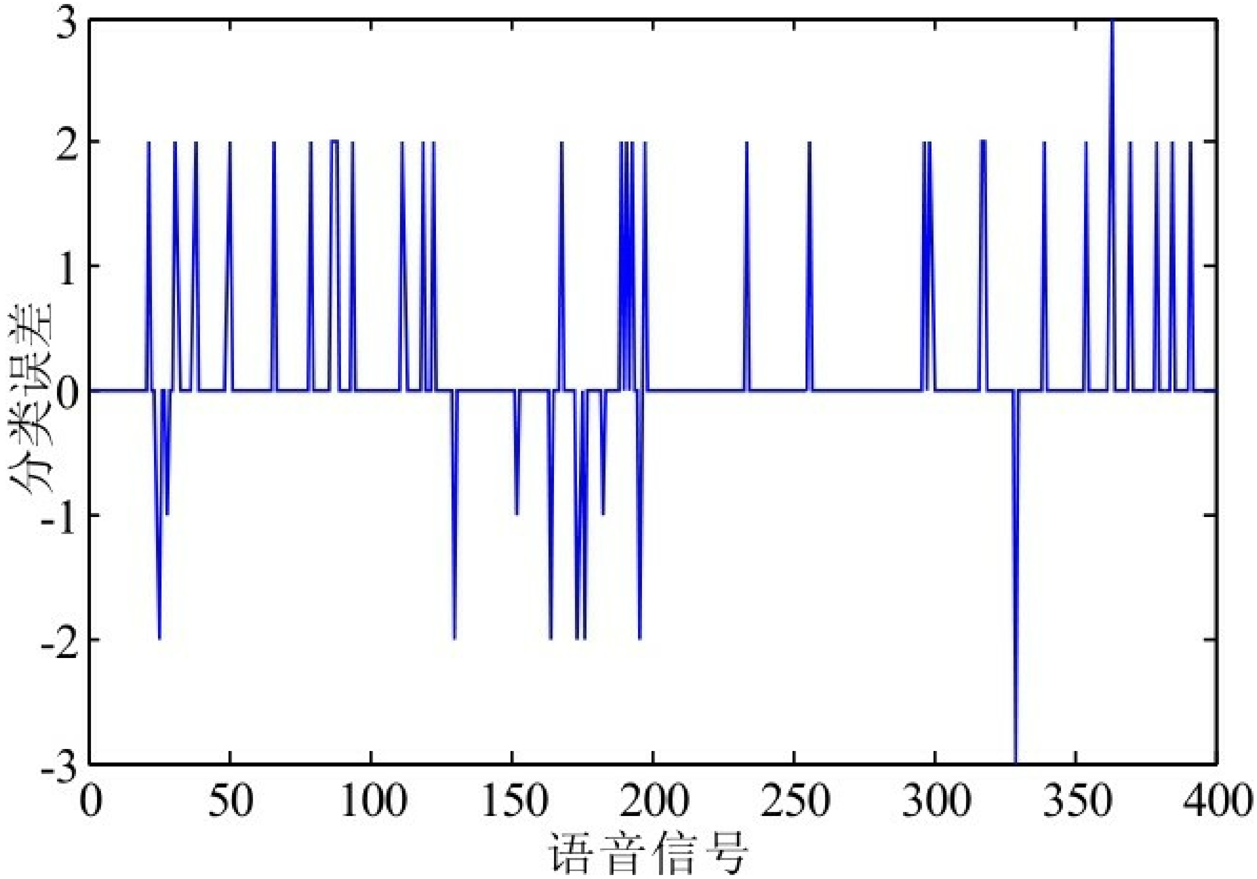


图7-29 语音分类误差曲线

程序运行结果如下。

rr =

0.6702   1.0000   0.9811   0.9175

从BP神经网络的分类结果可以得到，基于BP神经网络的语音信号分类算法的准确性还是比较高的，能够正确识别出信号所属类别。

7.4.4 BP神经网络的非线性函数拟合应用

在实际工程应用中，经常会遇到一些复杂的非线性系统，这些系统都有一个共同点：很难用数学方法建模。而BP神经网络可以准确表达出非线性系统。

如果把未知的非线性系统看作一个黑箱，首先可以用系统的输入和

输出数据训练　BP神经网络，使其能够准确表达出该系统的关系，然后就用训练好的网络预测系统的输出。

本节讨论应用BP神经系统实现对非线性函数的拟合。

【例7-7】假设非线性函数关系为

$$z = x^2 - y^2 + 3$$

使用BP神经网络实现对该函数的拟合。

解：首先需要获取BP神经网络的输入和输出数据，得到这些数据可以用如下MATLAB代码。

```
clear all
clc
x=rand(1,1500);
y=rand(1,1500);
p=[x;y]';
z=x.^2-y.^2+3;
```

该函数的图形如图7-30所示。

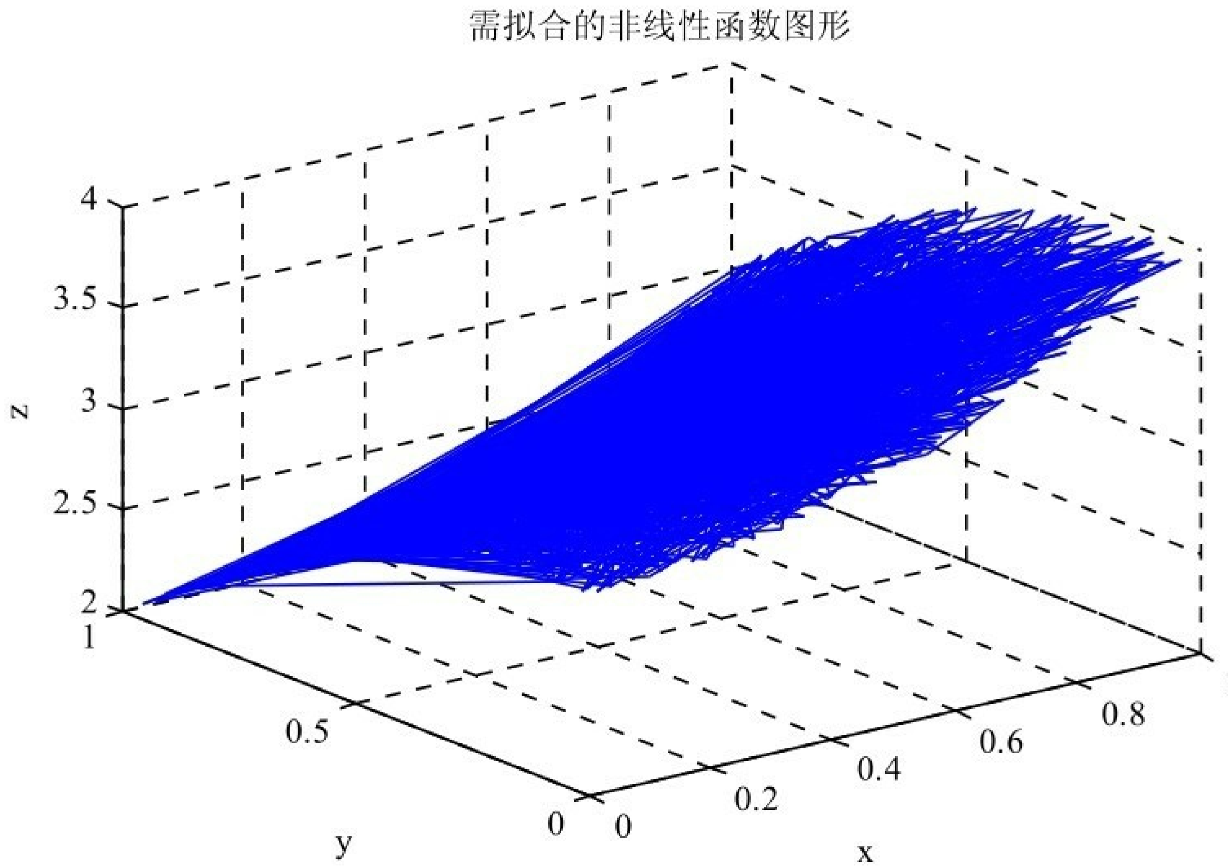


图7-30 需拟合的非线性函数图形

拟合函数的MATLAB代码如下。

```
clear all
clc
%% 训练数据预测数据提取及归一化
%下载输入输出数据
load data x y z p
%从1到1500间随机排序
k=rand(1,1500);
[m,n]=sort(k);
%找出训练数据和预测数据
input_train=p(n(1:1400),:)';
output_train=z(n(1:1400));
```

```
input_test=p(n(1401:1500),:)';
output_test=z(n(1401:1500));
%选连样本输入输出数据归一化
[inputn,inputps]=mapminmax(input_train);
[outputn,outputps]=mapminmax(output_train);
%% BP网络训练
%%初始化网络结构
net=newff(inputn,outputn,5);%%%%%%%%%%%%%%%%%%%
net.trainParam.epochs=500;
net.trainParam.lr=0.05;
net.trainParam.goal=0.0004;
%网络训练
net=train(net,inputn,outputn);
%% BP网络预测
%预测数据归一化
inputn_test=mapminmax('apply',input_test,inputps);
%网络预测输出
an=sim(net,inputn_test);
%网络输出反归一化
BPoutput=mapminmax('reverse',an,outputps);
%% 结果分析
figure (1)
plot(BPoutput,':og')
hold on
plot(output_test,'-*');
ylabel('仿真输出','fontsize',12)
xlabel('样本','fontsize',12)
```

```
%预测误差
error=BPoutput-output_test;
figure (2)
subplot(2,1,1)
plot(error,'-*')
ylabel('绝对误差','fontsize',12)
xlabel('样本','fontsize',12)
subplot(2,1,2)
plot((output_test-BPoutput)./BPoutput*100,'-*');
ylabel('误差百分比','fontsize',12)
xlabel('样本','fontsize',12)
errorsum=sum(abs(error))
```

运行代码得到BP神经网络仿真输出值与目标值比较示意图如图7-31所示。

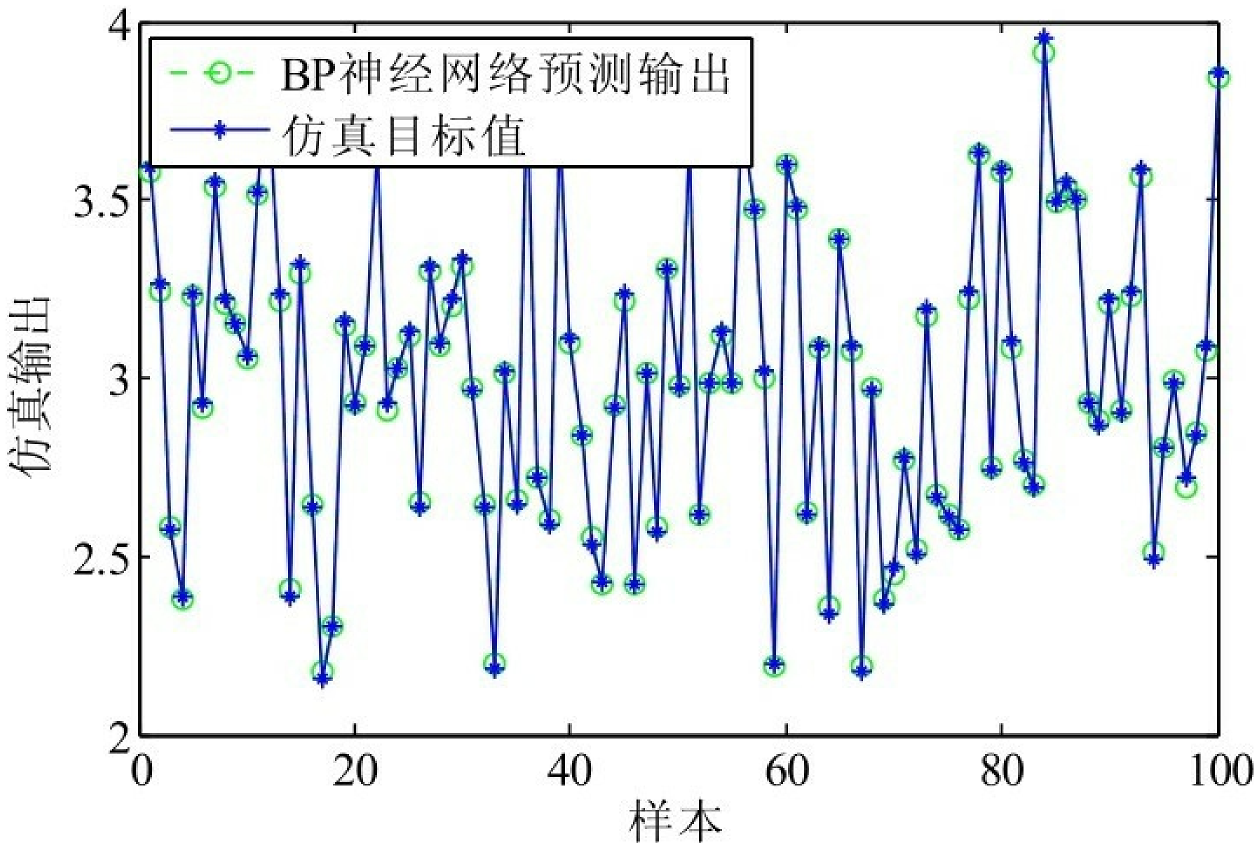


图7-31 BP神经网络仿真输出值与目标值比较示意图

BP神经网络训练的误差变化曲线图如图7-32所示。

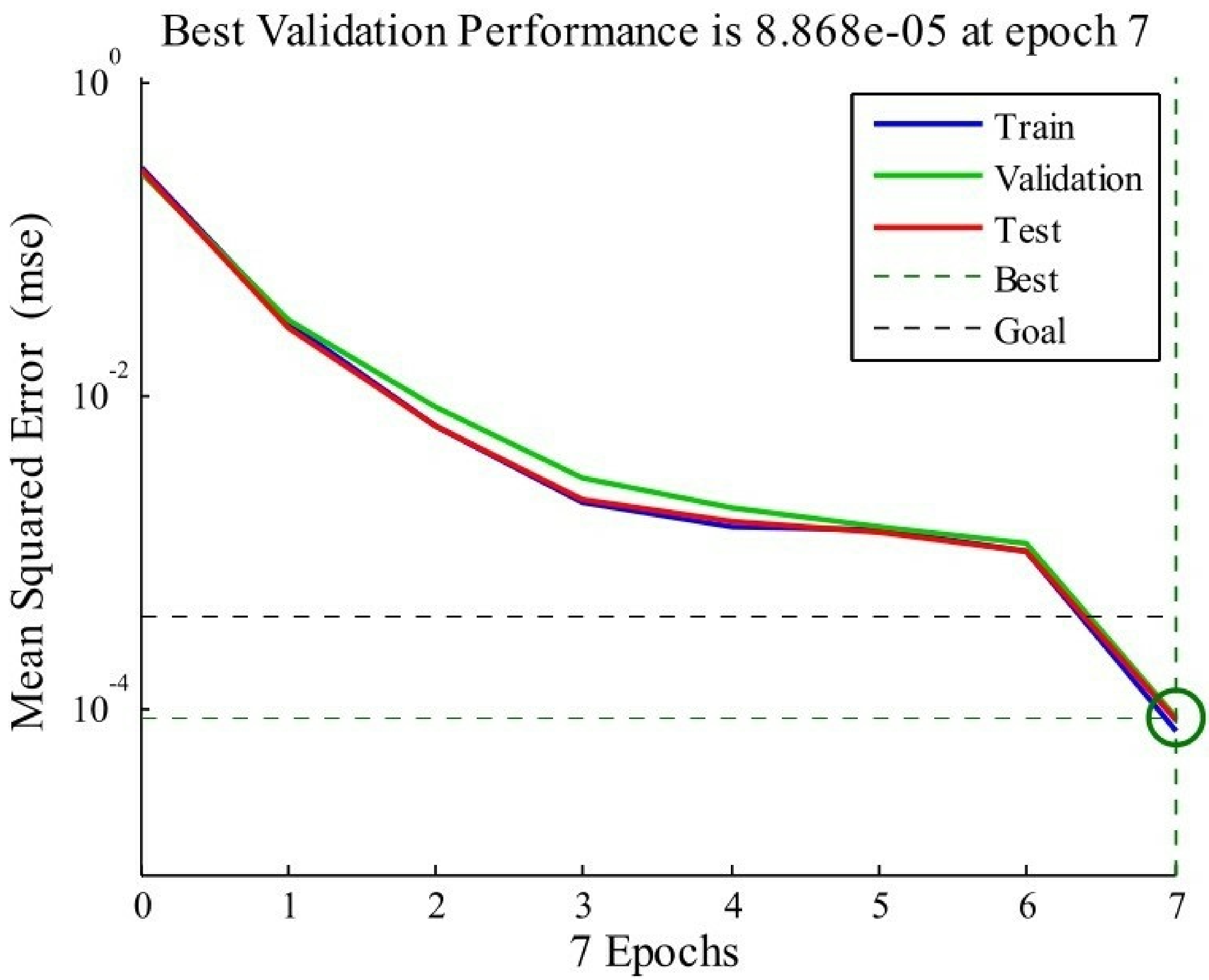


图7-32 BP神经网络训练的误差变化曲线图

用经过训练的 BP 神经网络预测函数的输出，预测结果误差和预测误差的百分比如图7-33所示。

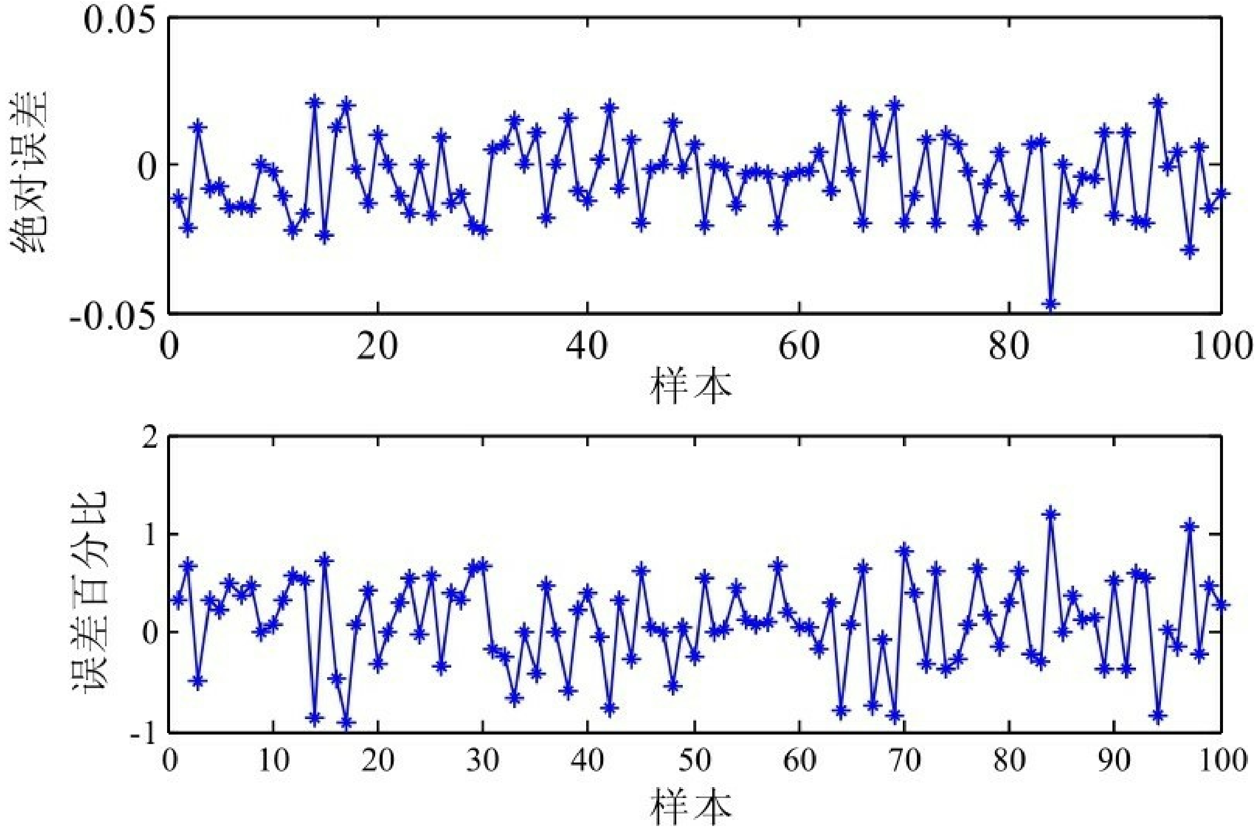


图7-33 BP神经网络预测结果误差

从图7-33可以看出，BP神经网络具有较高的拟合能力。

7.5 本章小结

BP神经网络可以用作分类、聚类、预测等。神经网络需要有一定量的历史数据，通过历史数据的训练，网络可以学习到数据中隐含的知识。本章重点介绍了 BP 神经网络的学习算法和训练，对一些常用的 BP 神经网络工具箱函数也做了简单介绍。最后通过几个案例，详细说明了BP神经网络的应用。

第8章 RBF神经网络

1985 年，Powell 提出了多变量插值的RBF 函数（Radical Basis Function，简称RBF）方法。1988年，Moody和Darken提出了一种神经网络结构，即RBF神经网络，属于前向神经网络类型，它能够以任意精度逼近任意连续函数，特别适合于解决分类问题。

本章将主要介绍RBF神经网络模型、工具箱函数及MATLAB应用实例。

学习目标：

·熟悉RBF 神经网络模型

·掌握RBF 神经网络工具箱函数的使用

·掌握RBF 神经网络MATLAB 应用实例

8.1 RBF网络模型

RBF网络的结构与多层前向网络类似，它是一种三层前向网络。

第一层为输入层，由信号源结点组成；第二层为隐含层，隐单元数量由所描述问题决定，隐单元的变换函数是对中心点径向对称且衰减的非负非线性函数；第三层为输出层，它对输入模式的作用作出响应。

从输入空间到隐含层空间的变换是非线性的，而从隐含层空间的输出层空间变换是线性的。

8.1.1 RBF神经网络模型

具有n个输入的RBF神经网络的神经元结构如图8-1所示。

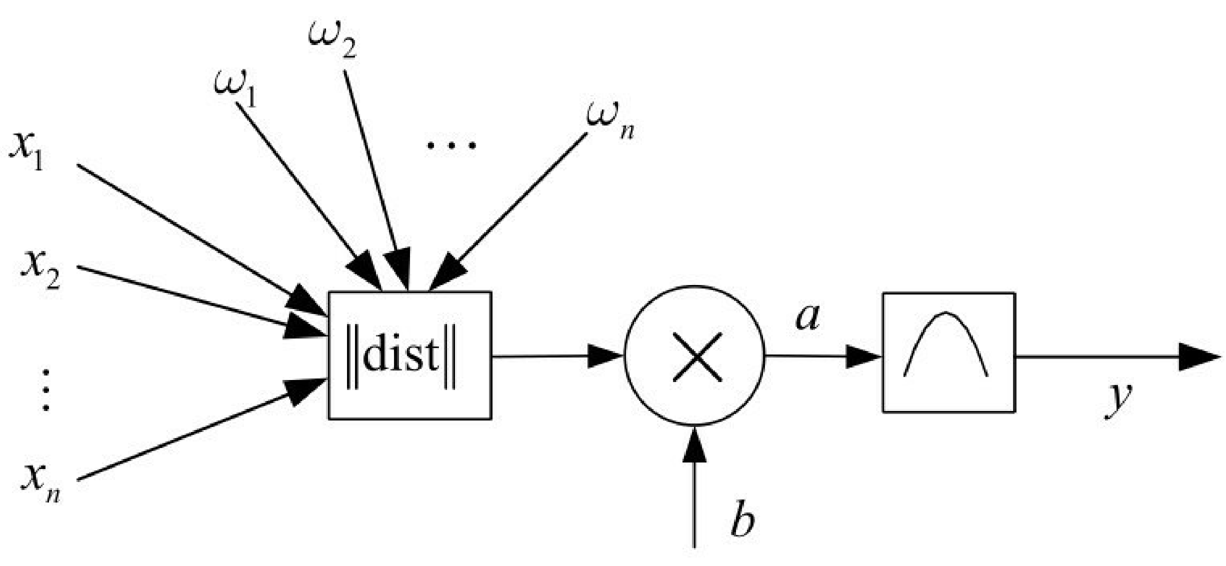


图8-1 RBF神经元模型

图8-1中的b为阈值，用于调整神经元的灵敏度。利用RBF神经元和线性神经元可以建立广义回归神经网络，该种神经网络适用于函数逼近方面的应用；RBF神经元和竞争神经元可以组建概率神经网络，此种神

经网络适用于解决分类问题。

RBF 神经网络的激活函数采用距离函数，并使用 RBF 函数（如Gaussian 函数）作为激活函数。RBF函数关于n维空间的一个中心点具有径向对称性，而且神经元的输入离该中心点越远，神经元的激活程度就越低。

在图8-1中，RBF神经网络的激活函数是以输入向量和权值向量之间的距离$\|\text{dist}\|$作为自变量的。RBF神经网络的激活函数的一般表达式为

$$R\left(\|\text{dist}\|\right)=e^{-\|\text{dist}\|^2}$$

由输入层、隐含层和输出层构成的一般RBF神经网络结构如图8-2所示。

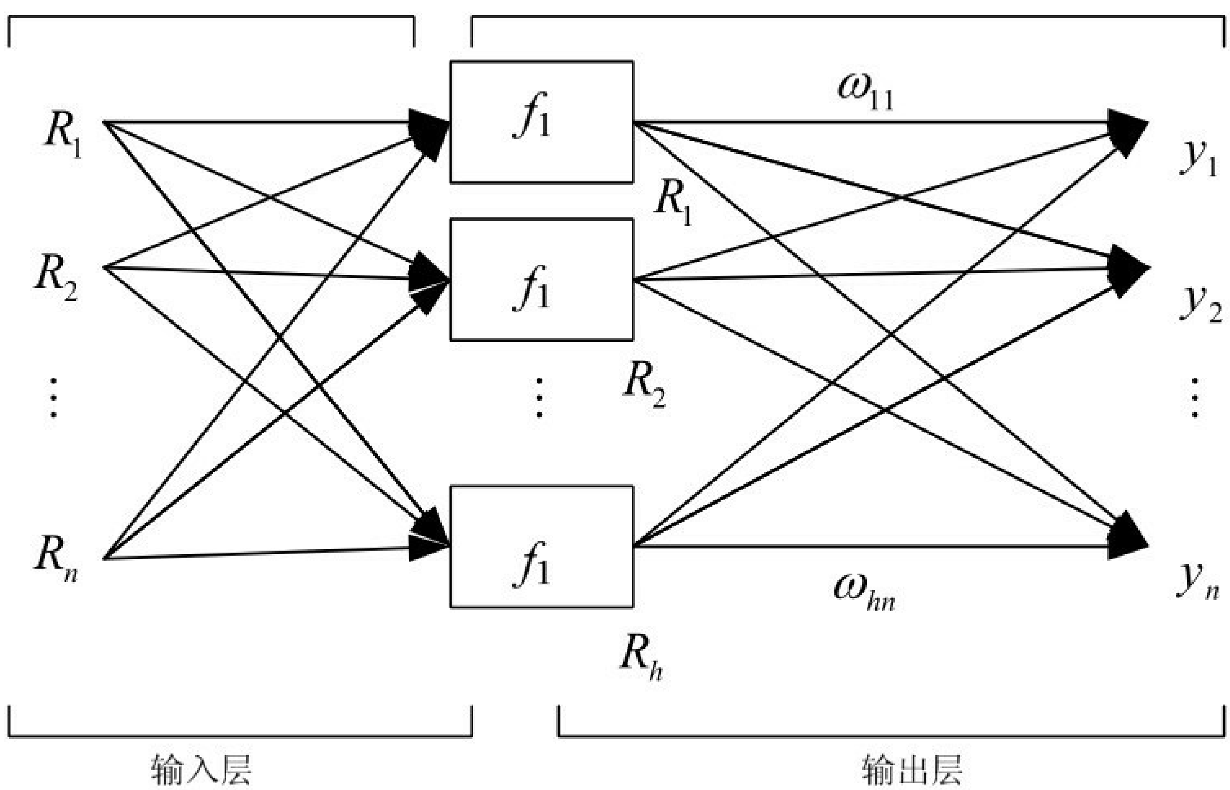


图8-2 RBF神经网络结构

在RBF网络中，输入层仅仅起到传输信号的作用，与前面所讲述的神经网络相比较，输入层和隐含层之间可以看作连接权值为1的连接。输出层和隐含层所完成的任务是不同的，因而它们的学习策略也不相同。

输出层是对线性权进行调整，采用的是线性优化策略。因而学习速度较快。而隐含层是对激活函数（格林函数或高斯函数，一般取高斯）的参数进行调整，采用的是非线性优化策略，因而学习速度较慢。

RBF神经网络的基本思想是，用RBF函数作为隐单元的“基”，构成隐含层空间，而输入层直接传递输入信号到隐含层，这样就可以将输入矢量直接（即不通过权值连接）映射到隐空间，使得隐含层单元输出即为网络输入的RBF函数映射（是非线性的）。

另一方面，由隐含层空间到输出空间的映射是线性的，即网络的输出是隐含层单元输出的线性加权和，此处的权即隐含层单元与输出层单元的连接权，是网络的可调参数。

由此可见，从总体上看，RBF网络由输入到输出的映射是非线性的，而网络输出对可调参数而言却又是线性的，这样网络的权值就可以用最小方差算法、递推最小二乘法等计算，从而大大加快了学习速度并避免了局部极小问题。

8.1.2 RBF网络的工作原理

当输入向量加到网络输入端时，RBF第一层的每个神经元都会输出一个值，代表输入向量和神经元权值向量之间的接近程度。

·如果输入向量与权值向量相差很多，则RBF第一层神经元的输出接近于0，经过第二层的线性神经元，输出也接近于0。

·如果输入向量与权值向量很接近，则RBF 第一层神经元的输出接近于1，经过第二层的线性神经元，输出值就更接近第二层权值。

在这个过程中，如果只有一个RBF神经元的输出为1，而其他神经元的输出均为0或者接近于 0，那么线性神经元的输出就相当于输出为 1 的神经元相对应的第二层神经元的权值。一般情况下，不止一个RBF神经元的输出为1，所以输出值也就会有所不同。

8.1.3 RBF 神经网络的具体实现

对RBF函数的定义如下。设X_0, X∈R^n（ R^n是n维实数空间），以X_0为中心，X 到X_0的距离为半径所构成的径向对称函数 $\{R(x)=R(\|X-X_0\|)\}$ 称为RBF函数。

对于RBF网络的数学描述可表达为，在n维空间中，给定n个输入样本$X_i\,(i=1,2,\ldots,n)$，则网络隐含层第k个节点的输出可以表示为

$$r_k=R(\|X_i-T_k\|)$$

其中，X_i是n维输入向量；$\|\bullet\|$通常为欧氏范数；T_k是第k个隐含层节点的中心(k=1, 2, ...,l)；

R(·)即 RBF 函数，具有局部感受特性，体现了 RBF 网络的非线性映射能力。网络输出层第j个节点的输出，则为隐含层节点输出的线性组合

$$y_i=\sum_{k=1}^{l}\omega_{kj}r_k-\theta_j$$

其中，ω_{kj}是隐含层到输出层的权值；θ_j是第j个输出节点的阈值。

高斯基函数具备如下优点。

·表示形式简单，即使对于多变量输入也不增加太多的复杂性。

·径向对称。

·光滑性好，任意阶导数都存在。

输出层节点j相应的输出则可以表示为

$$y_i = \sum_{k=1}^{l} \omega_{kj} r_k \left(j = 1, 2, \ldots, J \right)$$

由此可见，对于RBF网络来说， T_k,ω是最为重要的参数，设计RBF网络的任务就是用一定的学习算法来确定这3个参数。

8.2 RBF神经网络的学习算法

RBF神经网络的基本思想是，用RBF作为隐单元的“基”构成隐含层空间，这样就可以将输入矢量直接（即不需要通过权接）映射到隐空间。当RBF的中心点确定以后，这种映射关系也就确定了。而隐含层空间到输出空间的映射是线性的，即网络的输出是隐单元输出的线性加权和。此处的权即为网络可调参数。

由此可见，从总体上看，网络由输入到输出的映射是非线性的，而网络输出对可调参数而言却又是线性的。这样网络的权就可由线性方程直接解出，从而大大加快学习速度并避免局部极小问题。

RBF神经网络学习算法需要求解的参数有3个：基函数的中心、方差以及隐含层到输出层的权值。根据RBF函数中心选取方法的不同，RBF网络有多种学习方法，如随机选取中心法、自组织选取中心法、有监督选取中心法和正交最小二乘法等。

下面将介绍自组织选取中心的RBF神经网络学习法。此方法由两个阶段组成：一是自组织学习阶段，此阶段为无导师学习过程，求解隐含层基函数的中心与方法；二是有导师学习阶段，此阶段求解隐含层到输出层之间的权值。

RBF神经网络中常用的RBF函数是高斯函数，因此RBF神经网络的激活函数可表示为

$$R\left(x_p - c_i\right) = \exp\left(-\frac{1}{2\sigma^2}\left\|x_p - c_i\right\|^2\right) \qquad (1)$$

式中，$\left\|x_p - c_i\right\|$是欧式范数，c为高斯函数的中心，σ为高斯函数的方差。

RBF神经网络的输出为

$$y_j = \sum_{i=1}^{h} \omega_{ij} \exp\left(-\frac{1}{2\sigma^2} \left\| x_p - c_i \right\|^2 \right) \tag{2}$$

式中，$x_p = \left(x_1^p, x_2^p, \cdots, x_m^p \right)$是第p个输入样本， p=1,2,…,P表示样本总数，c_i为网络隐含层结点的中心，ω_{ij}为隐含层到输出层的连接权值，i=1, 2,", h为隐含层的节点数， y_j为与输入样本对应的网络的第j个输出结点的实际输出。

设斜d是样本的期望输出值，那么基函数的方差可表示为

$$\sigma = \frac{1}{p} \sum_{j}^{m} \left\| d_j - y_j c_i \right\|^2$$

学习算法具体步骤如下。

基于K-均值聚类方法求解基函数中心c。

（1）网络初始化。

随机选取h个训练样本作为聚类中心 $c_i \left(i = 1, 2, \cdots, h \right)$

（2）将输入的训练样本集合按最近邻规则分组。

按照 x_p与中心为c_i之间的欧式距离将 x_p分配到输入样本的各个聚类集合ϑ_p(p=1,2,…,P)中。

（3）重新调整聚类中心。

计算各个聚类集合ϑ_p中训练样本的平均值，即新的聚类中心c_i，如果新的聚类中心不再发生变化，则所得到的c_i即为RBF神经网络最终的基函数中心，否则返回（1），进入下一轮的中心求解。

（4）求解方差σ_i。

该RBF神经网络的基函数为高斯函数，因此方差σ_i可由下式求解。

$$\sigma_i = \frac{c_{\max}}{\sqrt{2h}} \qquad i = 1, 2, \cdots, h$$

式中， c_{max}为所选取中心之间的最大距离。

（5）计算隐含层和输出层之间的权值。

隐含层至输出层之间神经元的连接权值可以用最小二乘法直接计算得到，计算公式如下。

$$\omega = \exp\left(\frac{h}{c_{\max}^2}\left\|x_p - c_i\right\|^2\right) \qquad p = 1, 2, \cdots, P; i = 1, 2, \cdots, h$$

上述所说的方法是在网络结构尽可能简单（即隐含层单元数尽可能小）的前提下，通过优化RBF的参数，特别是RBF的中心来改善网络的性能，这些方法可以实现无限逼近效果，但计算起来较为复杂繁琐。

另外一种方法是用增加网络结构单元来实现，即中心固定增加隐含层的神经元。这时网络只有输出层权值W为一个自由参数，它可以用线性优化策略进行调整。这种方法可以同时实现网络结构（即隐含层单元数）和逼近精度的最佳组合，隐含层单元数根据用户的误差精度要求来对应设定。

但这种方法的效果与输入的样本数有关，在有限的样本的前提下，增加隐含层单元最多只能是达到样本的数目，再继续增加则不会改善网络的性能。

8.3 RBF网络工具箱函数

众所周知，BP网络用于函数逼近时，权值的调节采用的是负梯度下降法。这个调节权值的方法有局限性，即收敛慢和局部极小等。而RBF网络在逼近能力、分类能力和学习速度等方面均优于BP网络。

MATLAB中提供了4个RBF函数相关的函数，它们都是创建两层的神经网络，第一层都是RBF神经网络层，第二层是线性层或者竞争层。主要的区别是它们权值、阈值是否相同。

注意：RBF函数网络不需要训练，在创建的时候就自动完成训练。

8.3.1 RBF 工具箱函数

下面介绍几种常用的工具箱函数。

1.net = newrbe(P,T,spread)

newrbe()函数可以快速设计一个 RBF 函数网络，且使得设计误差为0。第一层神经元数目等于输入向量的个数，加权输入函数为dist，网络输入函数为netprod；第二层（线性层）神经元数模由输出向量T确定，加权输入函数为dotprod，网络输入函数为netsum。两层都有阈值。

第一层的权值初值为p'，阈值初值为0.8326/spread，目的是使加权输入为±spread时， RBF神经网络层输出为0.5，阈值的设置决定了每一个RBF神经元对输入向量产生响应的区域。

【例8-1】使用newrbe函数，设定输入和输出，建立一个RBF神经网络，当输入为5时，求RBF网络输出。

解：设定神经网络的输入和输出。

P = [1 2 3];

T = [2 4.1 5.9];

建立RBF神经网络。

net = newrbe(P,T);

当输入为5时，求RBF网络的输出。

P = 5;

Y = sim(net,P)

输出得到如下结果。

Y =

  1.7211

2.[net,tr] = newrb(P,T,goal,spread,MN,DF)

该函数和newrbe一样，只是可以自动增加网络的隐含层神经元数模直到均方差满足精度或者神经元数模达到最大为止。

【例8-2】使用newrb函数和例8-1中的输入和输出，建立一个RBF神经网络，当输入为5时，求RBF网络输出。

解：建立的MATALB代码如下。

P = [1 2 3];

T = [2.0 4.1 5.9];

net = newrb(P,T);

P = 5;

Y = sim(net,P)

在MATLAB软件Command Window窗口显示结果如下。

NEWRB, neurons = 0, MSE = 2.54

NEWRB, neurons = 2, MSE = 0

NEWRB, neurons = 3, MSE = 0

Y =

  6.1400

3.net = newgrnn(P,T,spread)

该函数用于建立广义回归网络。广义回归网络主要用于函数逼近。它的结构完全与newbre的相同，但是有以下几点区别。

·第二网络的权值初值为T。

·第二层没有阈值。

·第二层的权值输入函数为normpod，网络输入函数为netsum。

【例8-3】使用newgrnn函数和例8-1中的输入和输出，建立一个广义回归网络，当输入为5时，求RBF网络输出。

解：建立的MATALB代码如下。

```
P = [1 2 3];
T = [2.0 4.1 5.9];
net = newgrnn(P,T);
P = 5;
Y = sim(net,P)
```

在MATLAB软件Command Window窗口显示结果如下。

```
Y =
   5.8445
```

4.net = newpnn(P,T,spread)

该函数用于建立概率神经网络（PNN）。该网络与前面3个最大的区别在于，第二层不再是线性层而是竞争层，并且竞争层没有阈值，其他同 newrbe，故 PNN 网络主要用于解决分类问题。

为网络提供一输入向量后，首先，RBF神经网络层计算该输入向量同样本输入向量之间的距离||dist||，该层的输出为一个距离向量；竞争层接受距离向量为输入，计算每个模式出现的概率，通过竞争传递函数为概率最大的元素对应输出1，否则为0。

注意：由于第二层是竞争层，故输入/输出向量必须使用ind2vec/vec2ind函数进行转换，也就是将索引转换为向量或者向量转换为索引。

【例8-4】使用newpnn函数和例8-1中的输入和输出，建立一个概率神经网络，当输入为5时，求RBF网络输出。

解：建立的MATALB代码如下。

```
P = [1 2 3 4 5 6 7];
Tc = [1 2 3 2 2 3 1];
T = ind2vec(Tc);
net = newpnn(P,T);
p=5;
Y = sim(net,p)
```

在MATLAB软件Command Window窗口显示结果如下。

```
Y =
   0
   1
   0
```

8.3.2 转换函数

1.ind2vec

该函数用于将数据索引转换为向量组，调用格式如下所示。

```
vec=ind2vec(ind)
```

其中，ind表示数据索引列向量。

vec表示函数返回值，一个稀疏矩阵，每行只有一个1，矩阵的行数等于数据索引的个数，列数等于数据索引中的最大值。

通过下面一组代码演示ind2vec　的计算原理。在MATLAB　软件Command Window窗口输入如下语句。

```
>> ind = [1 3 2 3];
vec = ind2vec(ind)
```

得到结果如下所示。

ind =

1 3 2 3

vec =

(1,1) 1

(3,2) 1

(2,3) 1

(3,4) 1

可以看出，结果是一个4×3的矩阵，其中（x,y）是与数据索引列向量相对应的，x是数据索引值，y表示x在数据索引中的位置，由此组成了输出矩阵。

2.vec2ind

该函数用于将向量组转换为数据索引，其与ind2vec互逆。

例如在MATLAB 软件Command Window 窗口输入如下语句。

```
>> vec = [1 0 0 0; 0 0 1 0; 0 1 0 1];
>> ind = vec2ind(vec)
```

得到结果如下所示。

ind =

1 3 2 3

8.3.3 传递函数

RBF网络中的传递函数一般使用radbas。该函数的调用格式如下。

A = radbas(N,FP)

info=radbas(code)

其中，N为输入向量矩阵，A为函数返回矩阵，且与N一一对应，即N中的每个元素通过RBF函数得到A。

info=radbas(code)表示根据code值的不同返回有关函数的不同信息。包括以下几项。

·derive——返回导函数的名称。

·name——返回函数全称。

·output——返回输入范围。

·active——返回可用输入范围。

如果想查看传递函数波形，可以在Command Window窗口输入以下代码。

```
>> n =-5:0.1:5;
>> a = radbas(n);
>> plot(n,a)
```

得到传递函数波形如图8-3所示。

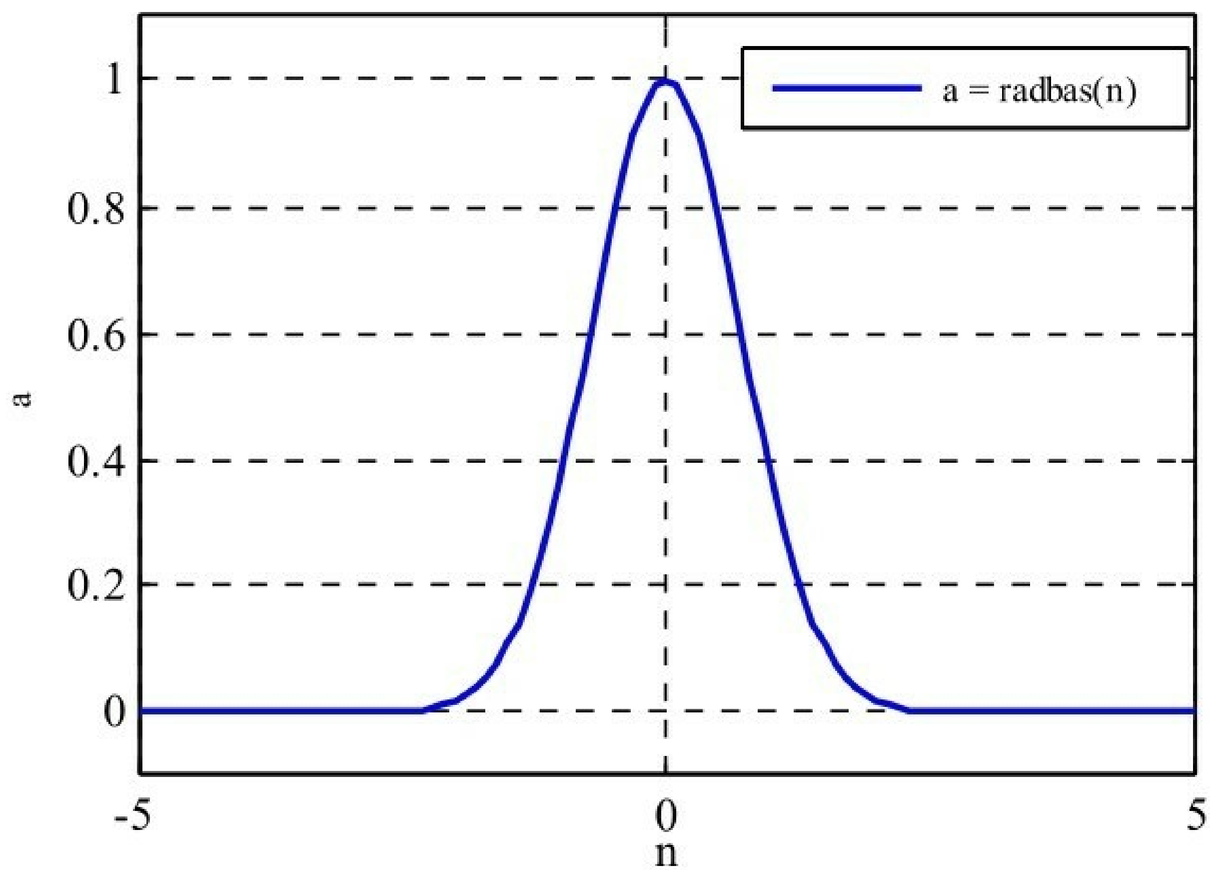


图8-3 传递函数radbas波形图

8.4 基于RBF网络的非线性滤波

自20世纪90年代中期以来，非线性估计理论出现了重大突破，人们抛弃了传统的非线性模型Taylor展开近似方法，采用计算简便、精度更高的插值多项式近似方法，甚至抛弃了对非线性模型进行线性变换的基本思想，而采用非线性变换思想。

本节主要介绍RBF网络的非线性滤波。

8.4.1 非线性滤波

早期的数字信号处理和数字图像处理主要以线性滤波器为主要处理手段。现在对一般的非线性滤波问题的研究相当活跃，常用的有同态滤波。

一般的非线性最优滤波可归结为求条件期望的问题。对于有限多个观测值的情形，条件期望原则上可以用贝叶斯公式来计算。但即使在比较简单的场合，这样得出的结果也是相当繁杂的，对实际应用或理论研究都很不方便。

与卡尔曼滤波类似，人们也希望能给出非线性滤波的某种递推算法或它所满足的随机微分方程。但一般它们并不存在，因此必须对所讨论的过程X与Y加以适当的限制。非线性滤波的研究工作相当活跃，它涉及随机过程论的许多近代成果，如随机过程一般理论、鞅、随机微分方程、点过程等。

其中一个十分重要的问题，是研究在什么条件下，存在一个鞅M，使得在任何时刻， M和Y都包含同样的信息；这样的M称为Y的信息过程。目前对于一类所谓“条件正态过程”，已经给出了非线性最优滤波的

可严格实现的递推算式。

在实际应用上，对非线性滤波问题往往采用各种线性近似的方法。

目前，最优非线性滤波存在“实时”问题，即滤波器权系数的实时计算；非线性滤波器的实时实现。描述系统的非线性差分方程如下。

$$\begin{cases} \text{状态方程}: x(n+1) = f(x(n)) + v(n) \\ \text{观测方程}: y(n+1) = h(x(n)) + w(n) \end{cases}$$

其中，f和h都是非线性函数，w(n)和v(n)均为零均值的白噪声序列。

8.4.2 RBF神经网络用于非线性滤波

仿真软件 MATLAB 中的神经网络工具箱为我们提供了一个新的运用神经网络的 GUI （图形用户界面）工具，使用这个工具可以对神经网络进行建立、初始化、训练和仿真等各种操作，界面友好直观。

在MATLAB的命令窗口输入nntool，回车，就可以打开GUI工具的主窗口。如图8-4所示。

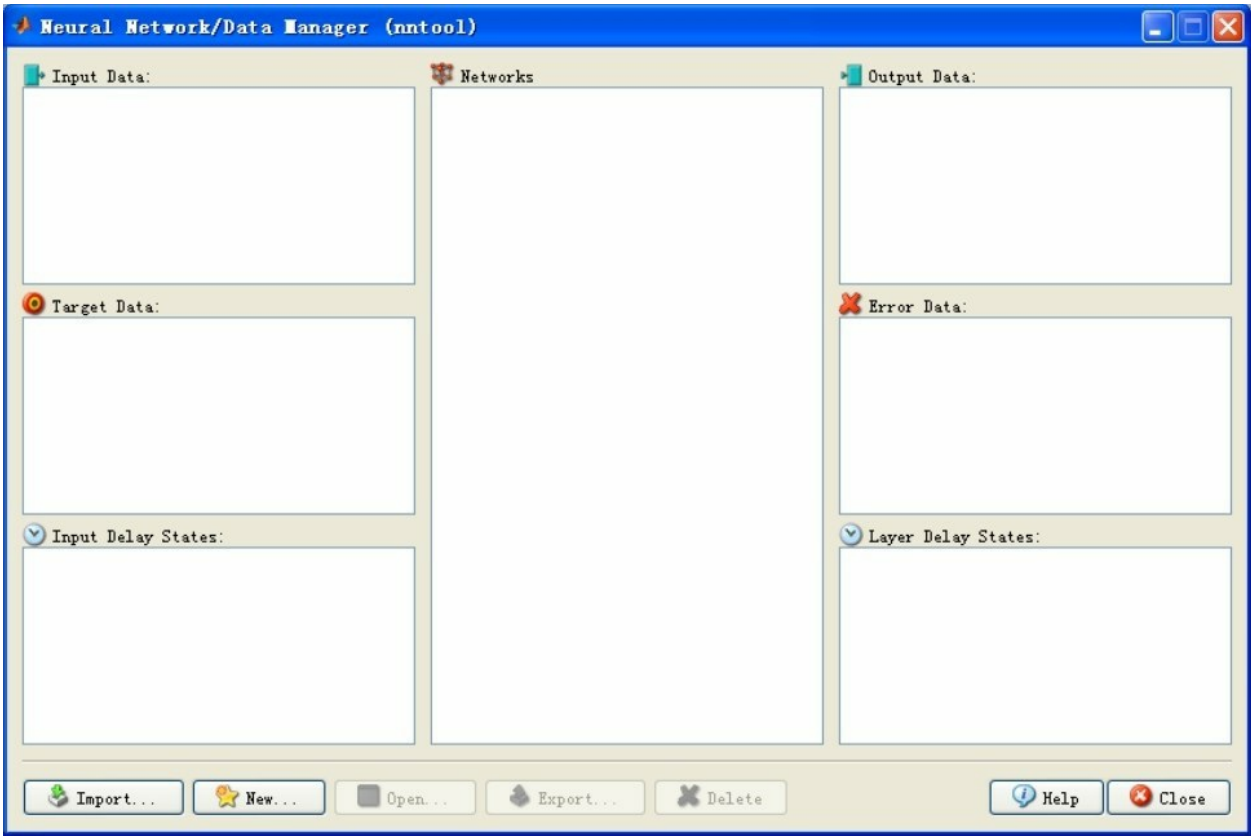


图8-4 GUI工具主窗口

下面通过对RBF算法进行滤波仿真。

假设发生故障的电流或者电压的函数为含有基波和3、5、7次谐波的函数，函数表达式如下。

$$y=\sin\left(2\cdot \mathrm{pi}\cdot 50\cdot t\right)+0.5\cdot\sin\left(2\cdot \mathrm{pi}\cdot 150\cdot t+\mathrm{pi}/6\right)$$

$$+0.4\cdot\sin\left(2\cdot \mathrm{pi}\cdot 250\cdot t+\mathrm{pi}/4\right)+0.3\cdot\sin\left(2\cdot \mathrm{pi}\cdot 350\cdot t+\mathrm{pi}/3\right)$$

上式表示的函数曲线如图 8-5 所示。从图 8-5 可见，故障情况下波形具有各次谐波，对于数据采集和计算带来误差，从而对故障的判断带来影响。因此采用RBF算法滤掉高次谐波。

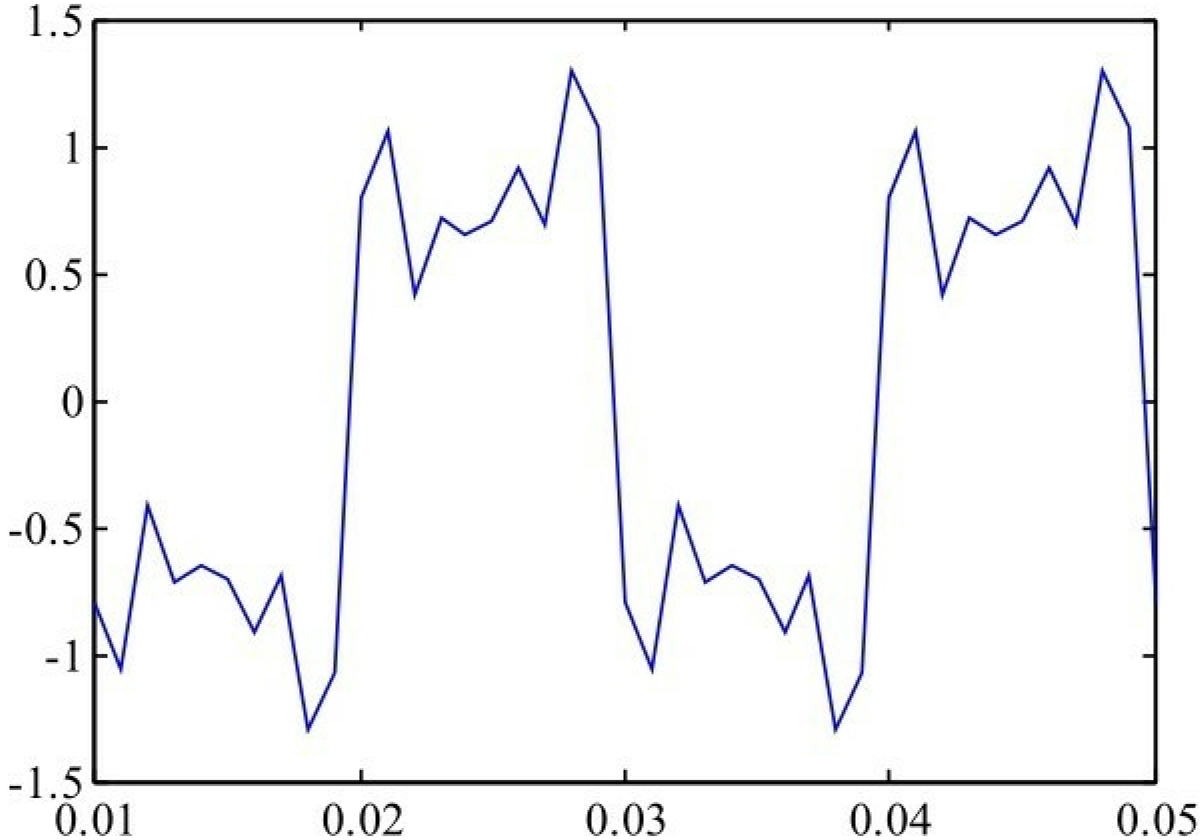


图8-5 含高次谐波的曲线图

RBF算法滤波仿真的程序设计如下。

（1）设定输入变量和目标变量。

在 MATLAB 的命令窗口定义需要的输入变量和目标变量，输入变量为如式y=sin(2·pi·50·t)+0.5·sin(2·pi·150·t+pi/6)+0.4·sin(2·pi·250·t+pi/4)+0.3·sin(2·pi·350·t+pi/3)中所示的含有 3、5、7 次高次谐波分量的模拟量，仿真的目的是滤波，所以目标变量为p=sin(2·pi·50·t)，具体定义如下。

```
>>clear all
>>clc
>>t=0:0.001:0.03;
>>y=sin(2*pi*50*t)+0.5*sin(2*pi*150*t+pi/6)+0.4*sin(2*pi*250*t+pi/4
>>p=sin(2*pi*50*t)
```

（2）导入输入和目标变量。

打开GUI工具窗口，单击图8-4中左下方的“Import”按钮，打开图8-6所示的变量样本导入窗口，分别将y和p作为输入变量和目标变量导入样本。

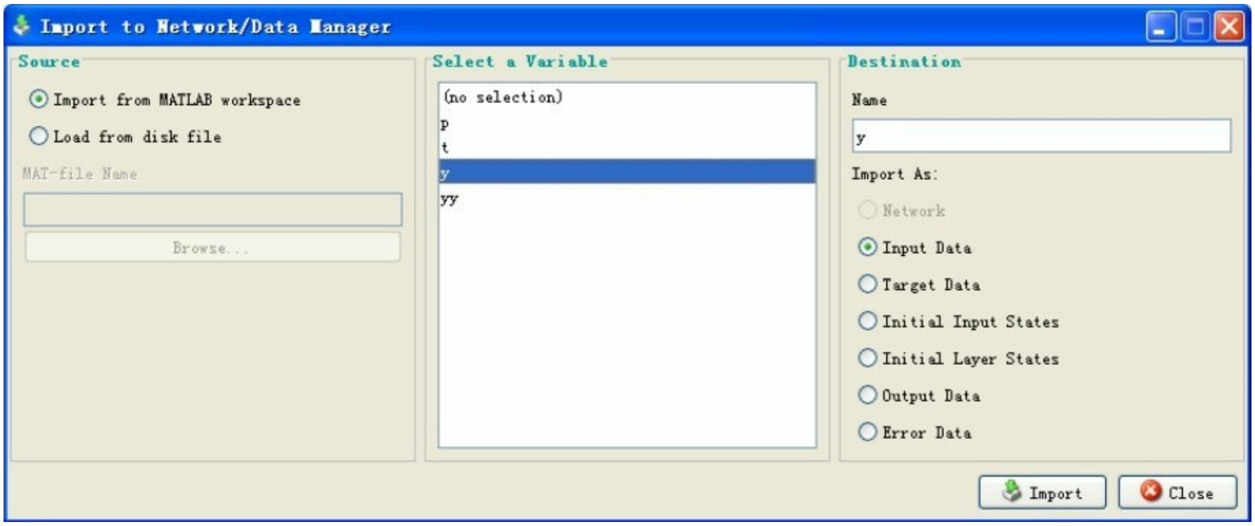


图8-6 变量样品导入窗口

当导入样本成功后，MATLAB会出现图8-7所示的提示框。

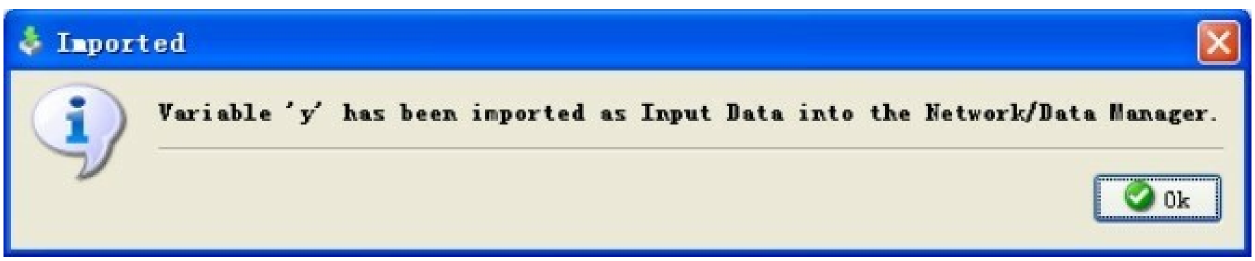


图8-7 样品导入成功提示框

（3）生成RBF网络。

回到GUI主窗口，单击“New”按钮，打开新建网络对话框，将新建网络名字命名为RBF，网络类型（Network Type）选择Radial basis(exact fit)，Spread constant 改为0.5，Input data 和Target data 分别选取y和p，如图8-8 所示。

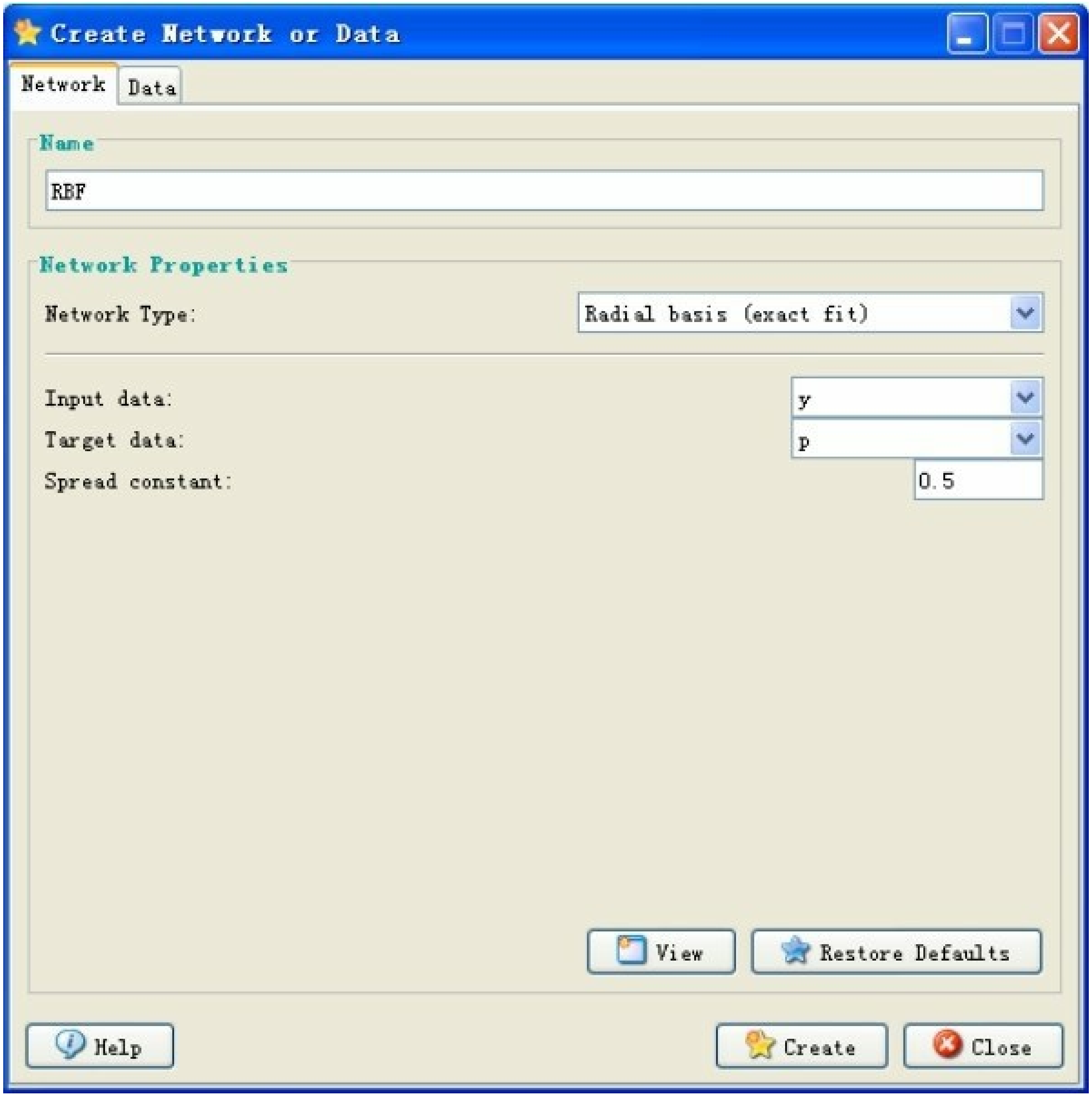


图8-8 新建网络对话框

单击图8-5中的“Create”按钮，生成RBF网络提示框如图8-9所示。

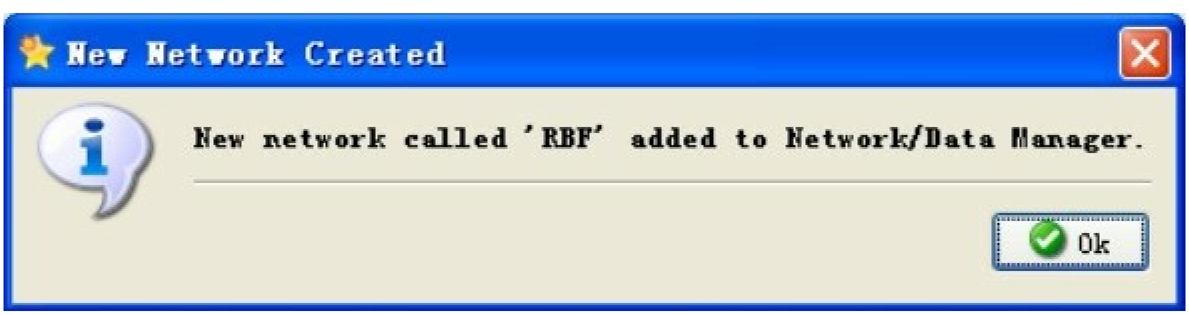


图8-9 生成RBF网络提示框

注意：在生成 RBF 网络后，MATLAB 软件已经完成神经网络训练和学习的过程，不需要另外再对RBF网络进行训练和学习。

回到GUI主窗口，选中Networks栏的网络名字RBF，单击“Open”按钮，打开网络结构图，如图8-10所示。

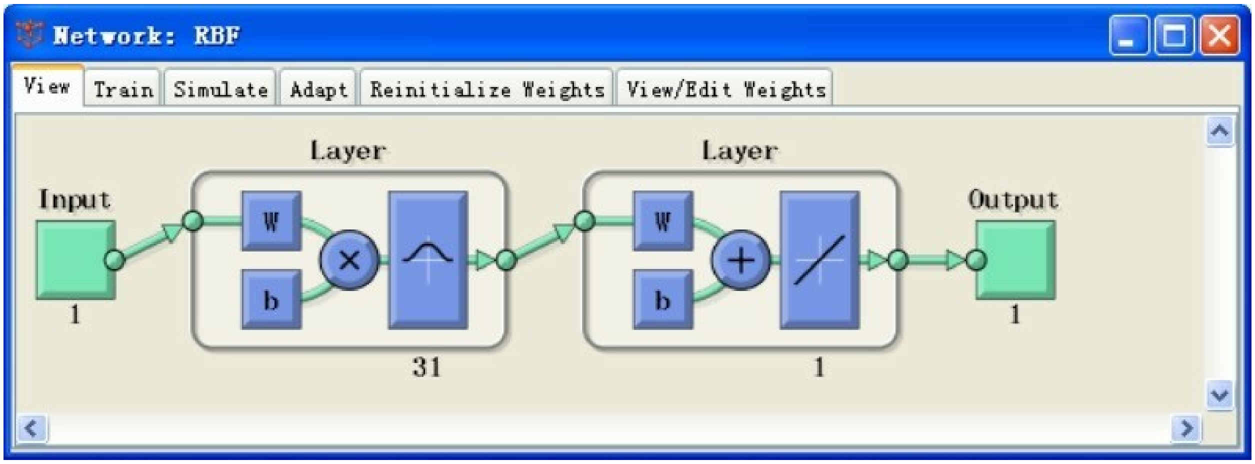


图8-10 RBF 网络结构图

（4）RBF网络仿真。

将变量y作为RBF网络仿真的输入变量。在RBF网络结构图中选择Simulate选项，在Inputs中选择变量yy作为RBF网络仿真的输入变量，Outputs保持默认配置，RBF网络结构图中Simulate选项设置如图8-11所示。

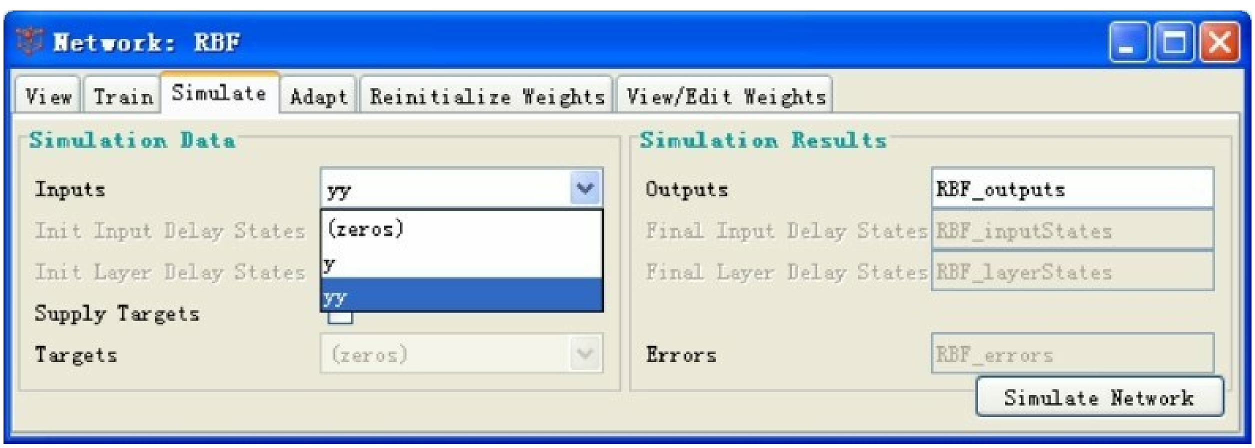


图8-11 RBF 网络结构图中Simulate选项设置示意图

单击图8-11中的“Simulate Network”按钮，MATLAB 软件进行RBF网络仿真，仿真结束后，出现图8-12所示提示框。

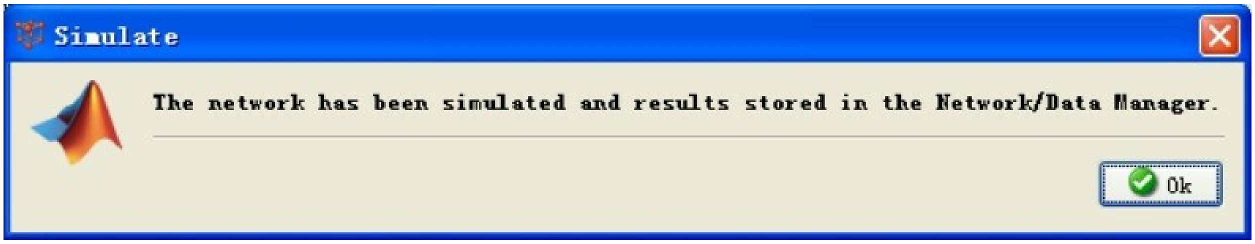


图8-12 RBF仿真结束提示框

仿真结束后，主窗口中Output Data选项下出现RBF仿真得到的输出变量RBF_outputs，如图8-13所示。

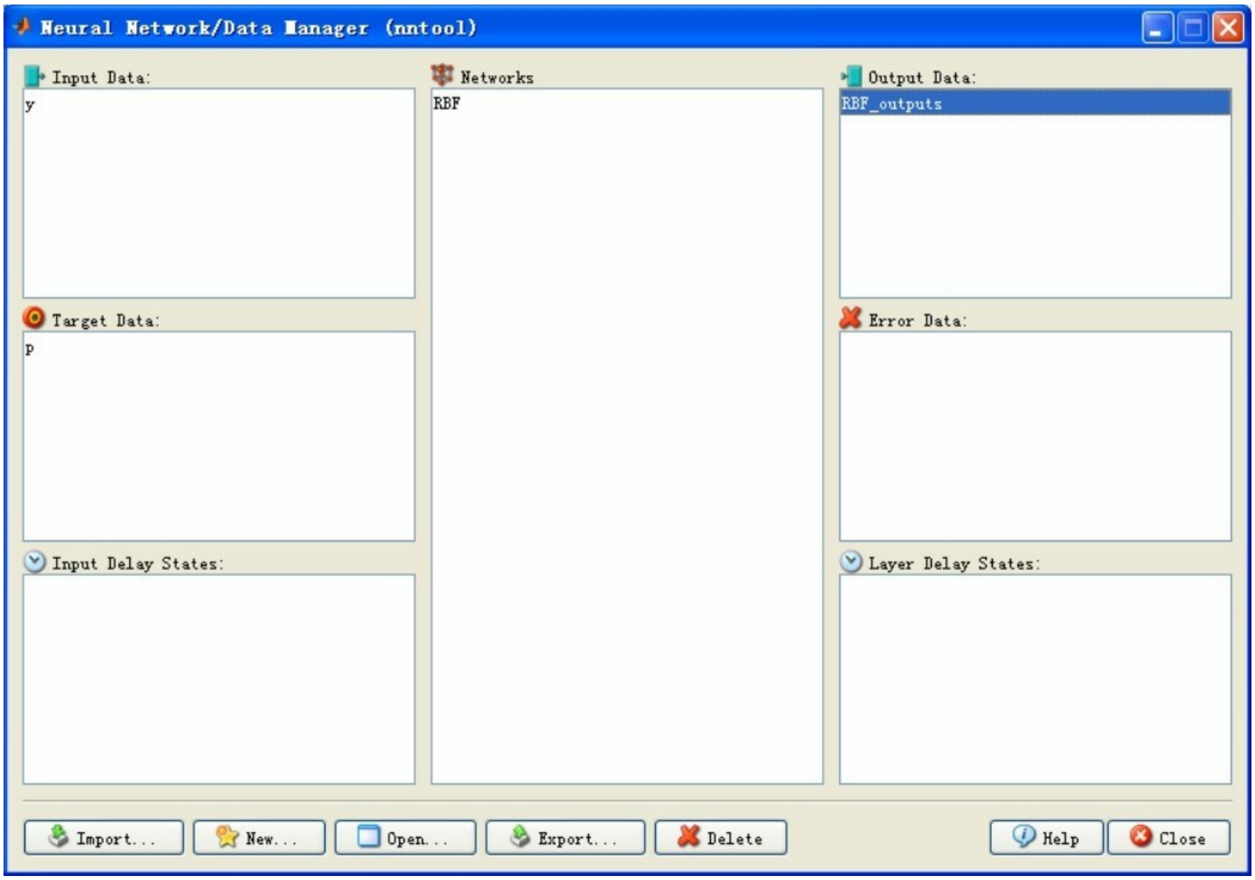


图8-13 显示仿真输出变量的主窗口

（5）导出仿真结果到Workspace窗口。

选中图8-13中RBF_outputs后，单击“Export”按钮，出现图8-14所示的包含仿真结果的Data Manager 窗口。

选中图8-14中RBF_outputs后，单击“Export”按钮，出现如图8-15所示的导出成功提示框。

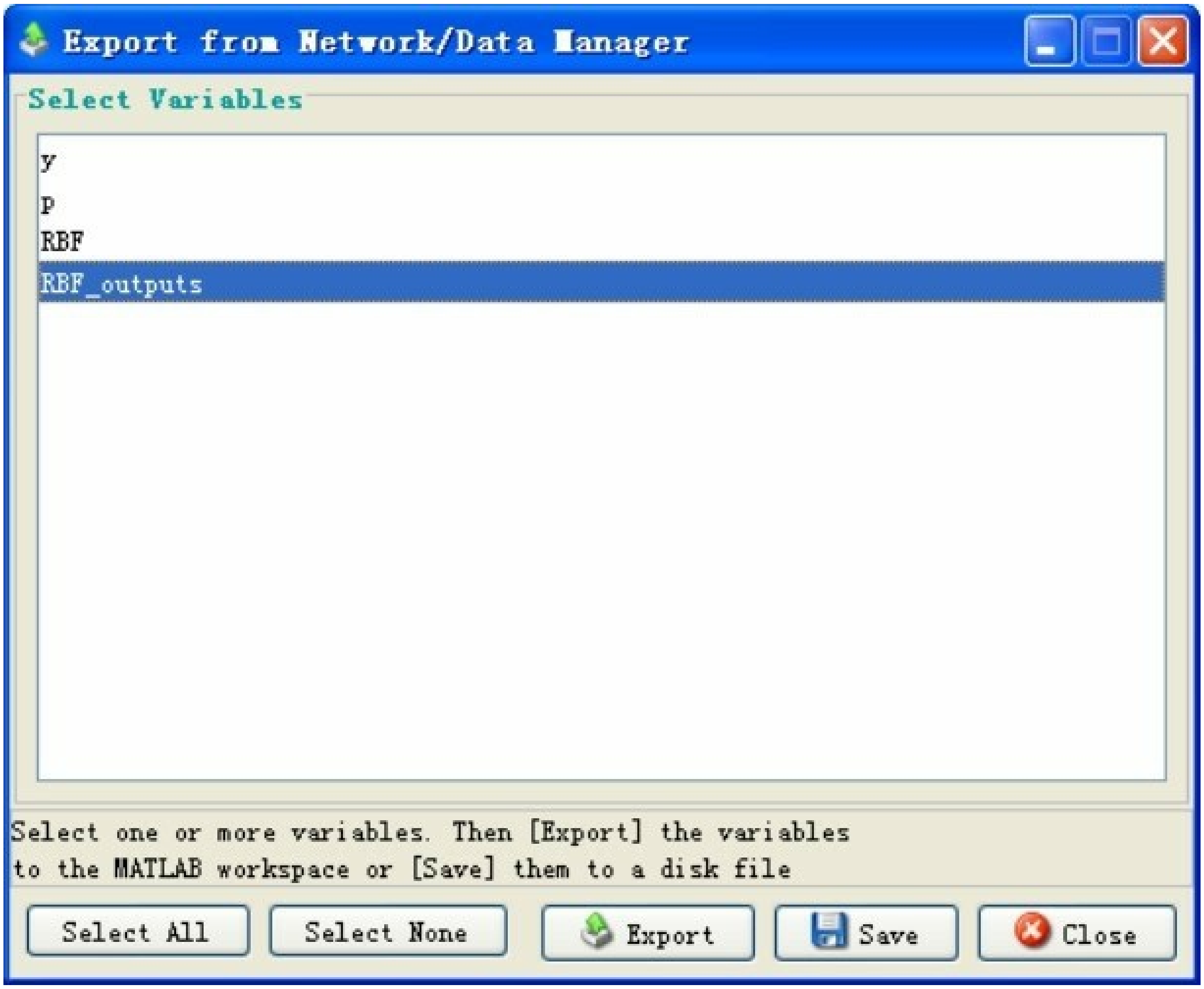


图8-14 包含仿真结果的Data Manager窗口

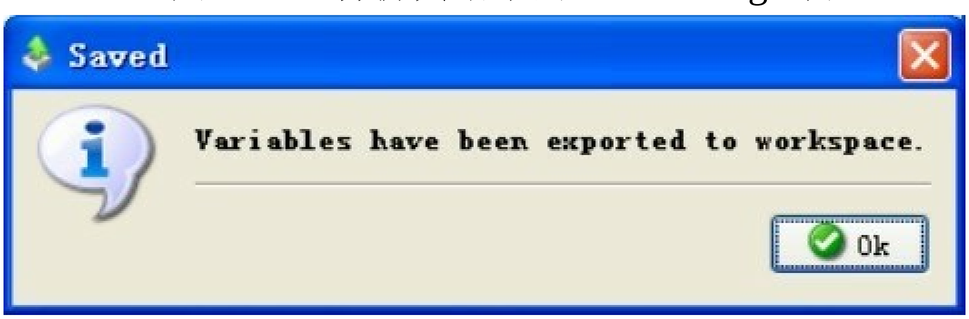


图8-15 导出成功提示框

（6）绘制目标结果与预测结果比较图。

在MATLAB命令窗口中输入命令。

```
>>plot(t,p,t, RBF_outputs);
```

可以看到目标结果和预测结果的比较图，如图8-16所示。

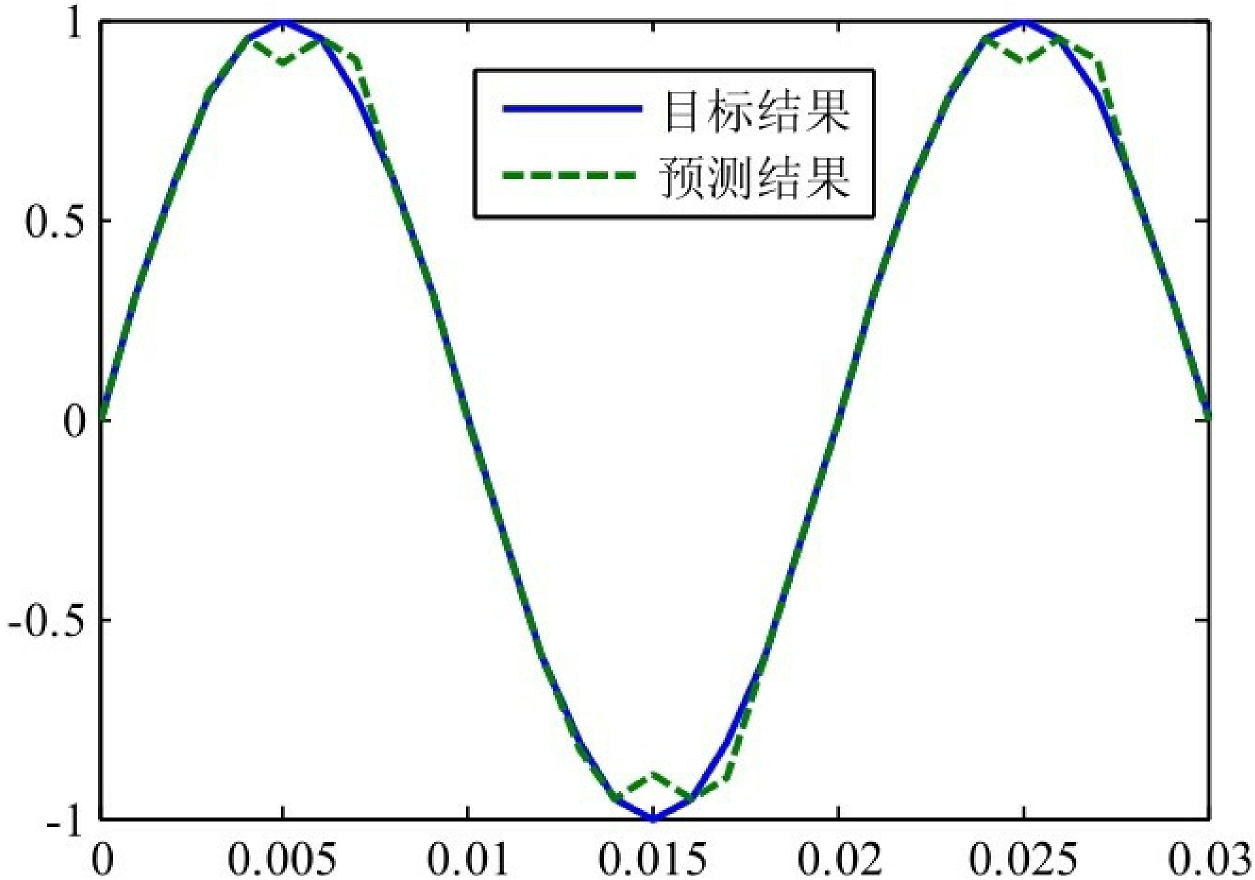


图8-16 目标结果和预测结果的比较图

从该图中可以看出，RBF神经网络在进行非线性滤波时得出的预测结果与目标结果曲线几乎完全重合，滤波效果非常明显。经过RBF算法滤波后，高次谐波被滤掉，这样有利于数据采集和计算，从而可以有效识别故障，保护正确的动作。

通过RBF神经网络算法的滤波仿真，可以得出结论，RBF神经网络是局部逼近型网络，能以任意精度逼近给定的非线性映射，可以避免算法中繁琐、冗长的计算以及局部最小陷阱的困扰，而且RBF网络的速度很快，隐含层节点的数目也在训练过程中确定，因此可以得到最优解。

8.5 RBF网络MATLAB应用实例

RBF网络具有良好的推广能力，在用于对具有复杂函数关系的问题作泛函逼近时，具有较高的精确度，而且由于网络本身的结构特点，学习速度比常规的 BP 算法要快得多。经过上面几节的介绍，本节以应用实例说明RBF神经网络的MATLAB程序设计。

曲线拟合是用连续曲线近似地刻画或比拟平面上离散点组所表示的坐标之间的函数关系的一种数据处理方法，是用解析表达式比较离散数据的一种方法。

RBF网络在完成函数逼近任务时速度最快，结构也比较简单。

【例8-5】设计一组系统输入x和输出y变量，建立RBF和BP神经网络，分别进行y=f(x)的曲线拟合，并比较二者的仿真时间。

解：设计系统输入x和输出y变量的程序如下所示。

```
x=0.1:0.1:1;
y=[0.3 -0.5 -1 -0.4 -0.9 0.9 -0.3 1.5 0.3 -1.1];
```

首先设计一个BP神经网络。以隐含层节点数为10的单输入和单输出两层网络进行曲线拟合。

创建和训练BP神经网络的MATLAB程序如下所示。

```
net = newff(minmax(x),[10,1],{'tansig' 'purelin'},'trainlm');
net.trainParam.epochs=2000;    %网络训练时间设置为2000
net.trainParam.goal=0.5;    %网络训练精度设置为0.5
net=train(net,y,x);        %开始训练网络
yBp=sim(net,x);        %仿真
figure(1);
plot(x,y,'-',x,yBp, '--')
```

```
title('训练后网络的输出结果');
xlabel('时间');
ylabel('仿真输出');
```

BP神经网络训练结果如图8-17所示。

运行程序得到输出结果如图 8-18 所示。BP 网络从生成网络到训练完成所需要的时间为2.84。

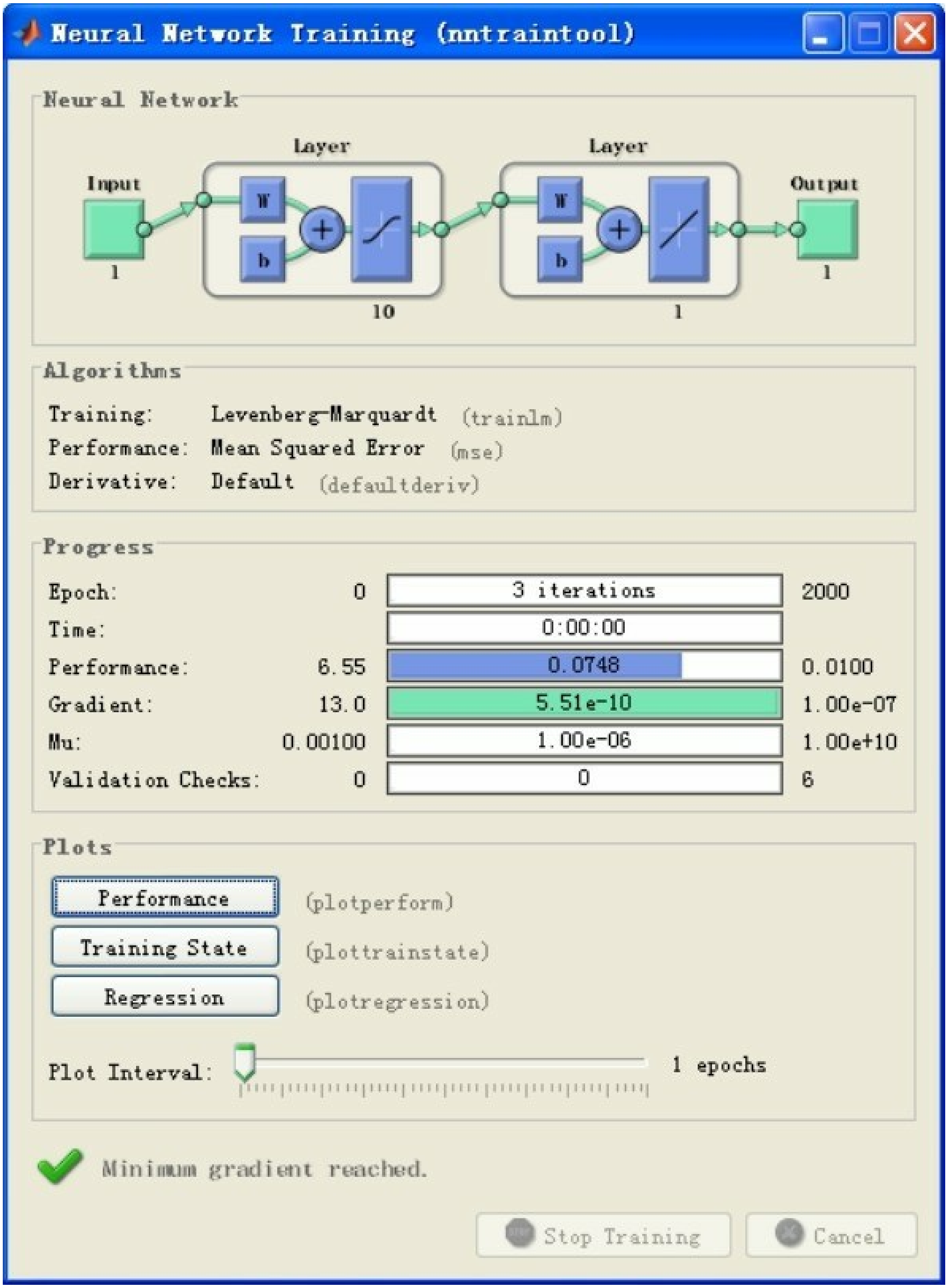


图8-17 BP神经网络训练结果

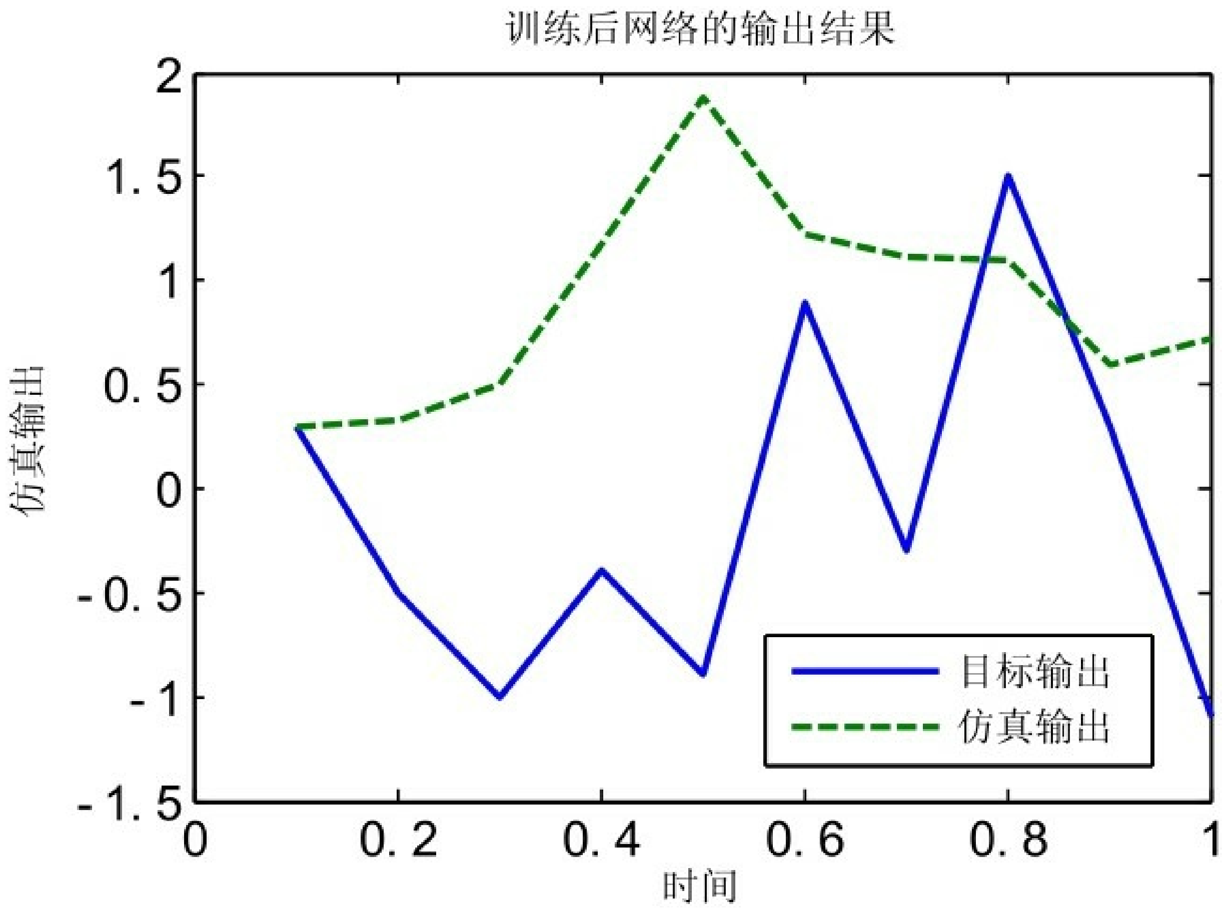


图8-18 BP神经网络输出结果

提示：由于 BP 神经网络建立的时候，随机初始化权值和阈值，所以每次运行程序的结果不一定相同，不同的计算机也将影响网络设计的时间。

创建和训练RBF神经网络的MATLAB程序如下所示。

```
% 建立RBF神经网络及仿真
t2=clock;
net=newrb(x,y,0.1,0.1,20,5);
data2=etime(clock,t2)
yRBF=sim(net,x);        %仿真
figure(2);
plot(x,y,'-',x,yRBF, '--')
```

title('训练后网络的输出结果');

xlabel('时间');

ylabel('仿真输出');

RBF神经网络训练误差曲线如图8-19所示。

运行程序得到输出结果如图 8-20 所示。BP 网络从生成网络到训练完成所需要的时间为1.32。

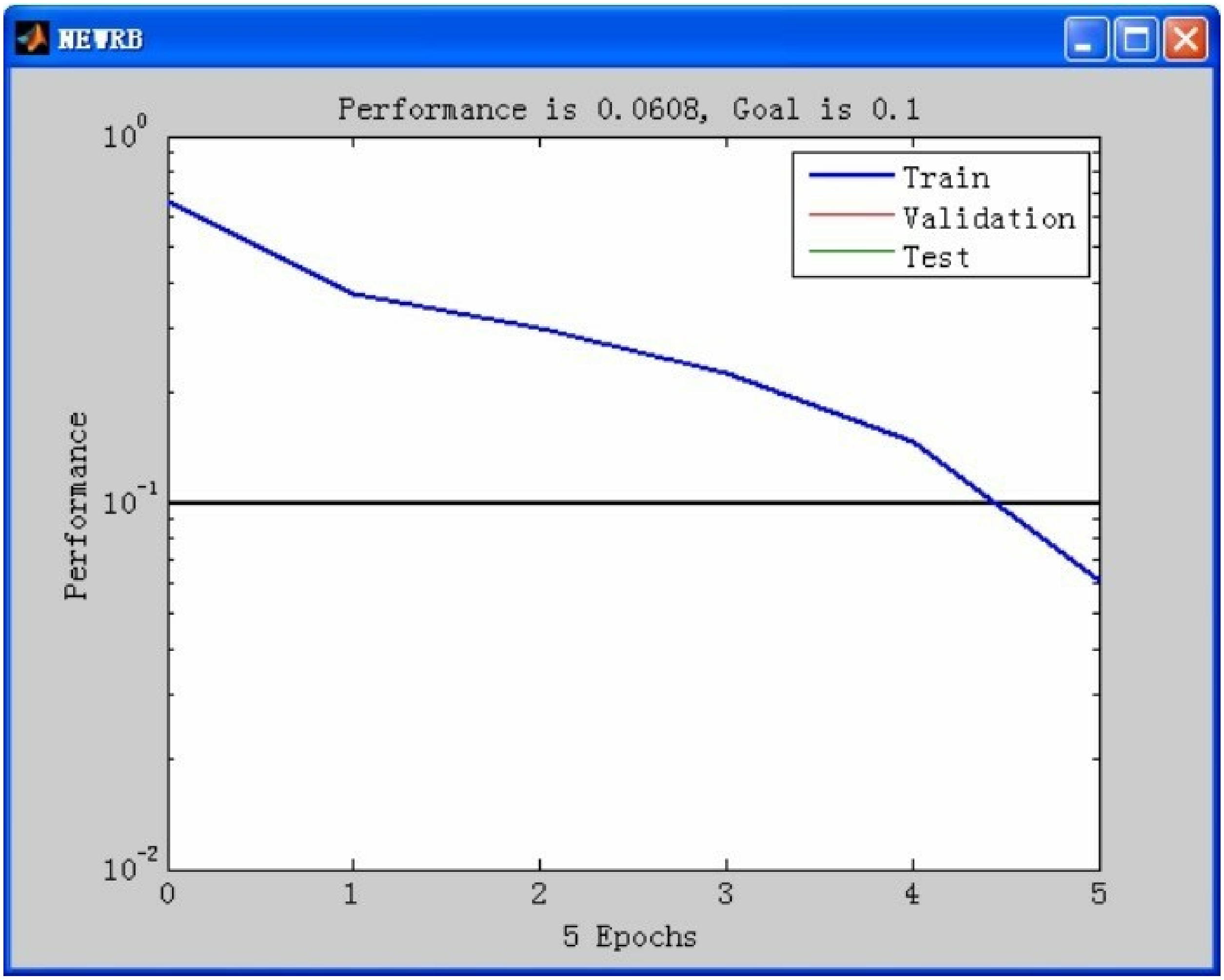


图8-19 RBF 神经网络训练误差曲线

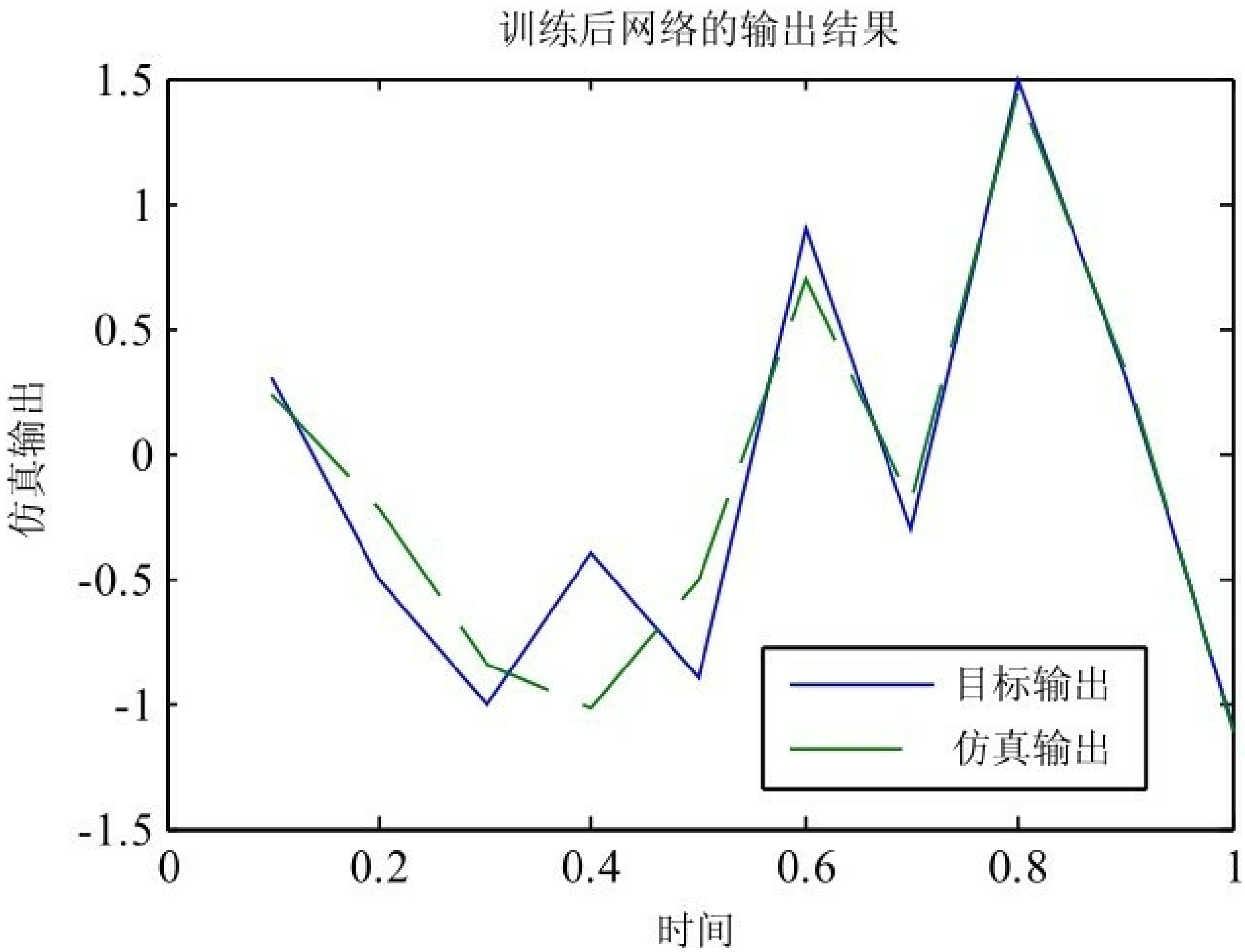


图8-20 RBF神经网络输出结果

可以看出，RBF神经网络仿真输出更加准确，建立网络和训练网络时间也比BP神经网络少。

【例8-6】用RBF网络拟合未知函数

$$y = 30 + x_1^2 - 5\cos(2\pi x_1) + 3x_2^2 - 5\cos(2\pi x_2)$$

。

解：本例首先随机产生输入变量 x_1，x_2，并根据产生的输入变量和未知函数求得输出变量y。根据输入变量x_1，x_2和输出变量y作为RBF网络的输入数据和输出数据，建立近似和精确RBF网络进行回归分析，并评价拟合效果。

使用exaceRBF网络来实现非线性的函数回归MATLAB如下所示。

```
clc
clear all
%%%%%%%%产生输入变量x1 x2%%%%%%%%%%%%%%%
x1=-1:0.01:1;
x2=-1:0.01:1;
%%%%%%%%产生输出变量y%%%%%%%%%%%%%%%%%%
y=30+x1.^2-5*cos(2*pi*x1)+3*x2.^2-5*cos(2*pi*x2);
%%%%%%%%建立RBF网
络%%%%%%%%%%%%%%%%%%%%
net=newrbe([x1;x2],y)
%%%%%%%%网络仿
真%%%%%%%%%%%%%%%%%%%%%%%
t=sim(net,[x1;x2]);
%%%%%%%%绘制拟合效果图%%%%%%%%%%%%%%%%%
figure(1)
plot3(x1,x2,y,'rd');
hold on;
plot3(x1,x2,t,'b-.');
view(100,25)
title(' RBF神经网络的拟合效果')
xlabel('x1')
ylabel('x2')
zlabel('y')
grid on
```

代码运行时，在MATLAB 的Command Windows 窗口会显示程序建立RBF 网络的相关信息如下所示。

```
net =
```

```
Neural Network
    name: 'Radial Basis Network, Exact'
    efficiency: .cacheDelayedInputs, .flattenTime,.memoryReduction
    userdata: (your custom info)
dimensions:
    numInputs: 1
    numLayers: 2
    numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
    numFeedbackDelays: 0
    numWeightElements: 805
    sampleTime: 1
connections:
    biasConnect: [1; 1]
    inputConnect: [1; 0]
    layerConnect: [0 0; 1 0]
    outputConnect: [0 1]
subobjects:
    inputs:{1x1 cell array of 1 input}
    layers:{2x1 cell array of 2 layers}
    outputs:{1x2 cell array of 1 output}
    biases:{2x1 cell array of 2 biases}
    inputWeights:{2x1 cell array of 1 weight}
    layerWeights:{2x2 cell array of 1 weight}
functions:
    adaptFcn: (none)
```

```
        adaptParam: (none)
        derivFcn: 'defaultderiv'
        divideFcn: (none)
        divideParam: (none)
        divideMode: 'sample'
        initFcn: 'initlay'
        performFcn: 'mse'
        performParam: .regularization, .normalization
        plotFcns:{}
        plotParams:{1x0 cell array of 0 params}
        trainFcn: (none)
        trainParam: (none)
    weight and bias values:
        IW:{2x1 cell} containing 1 input weight matrix
        LW:{2x2 cell} containing 1 layer weight matrix
        b:{2x1 cell} containing 2 bias vectors
    methods:
        adapt: Learn while in continuous use
        configure: Configure inputs & outputs
        gensim: Generate Simulink model
        init: Initialize weights & biases
        perform: Calculate performance
        sim: Evaluate network outputs given inputs
        train: Train network with examples
        view: View diagram
        unconfigure: Unconfigure inputs & outputs
```

仿真输出结果如图8-21所示。

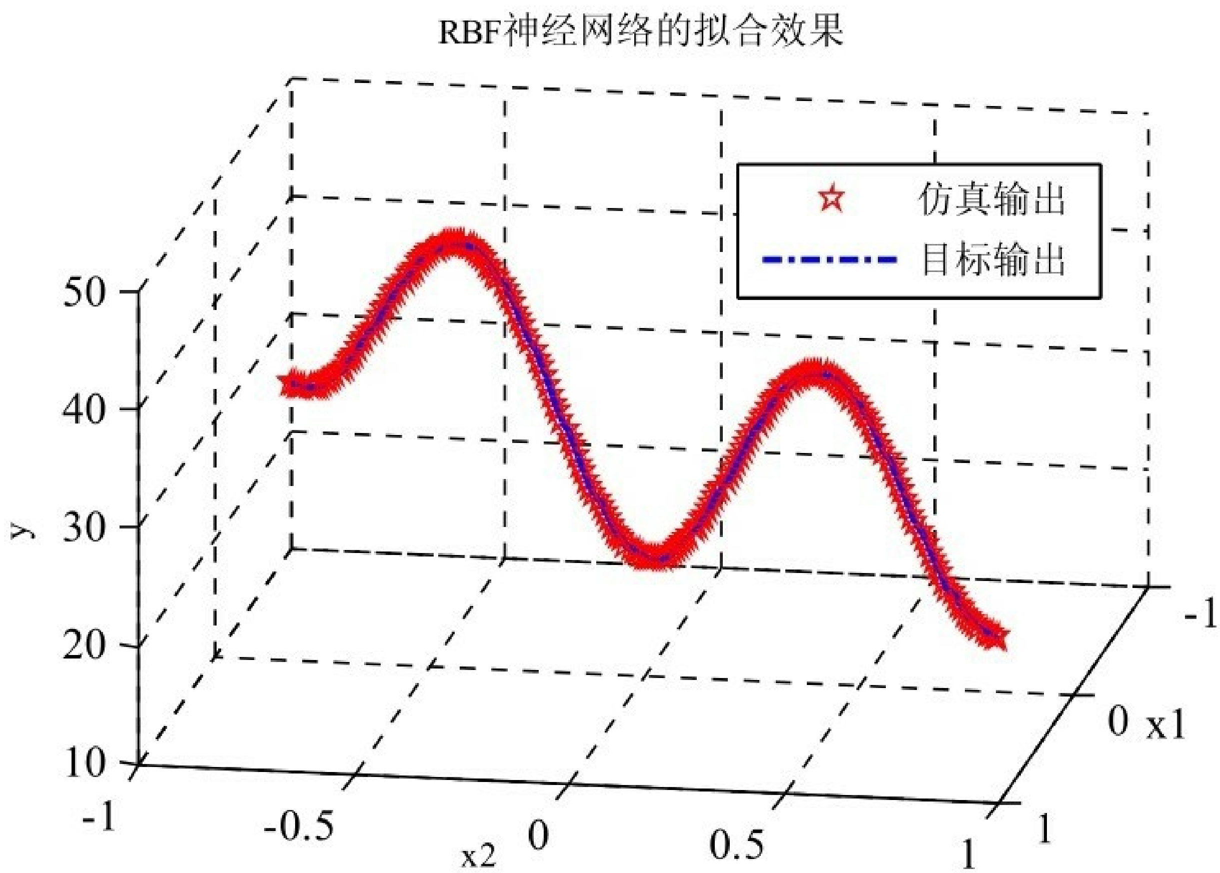


图8-21 exaceRBF网络来实现非线性的函数回归仿真结果

下面再用approximateRBF神经网络对例8-6中的函数进行拟合。其MATLAB程序如下所示。

```
clc
clear all
%%%%%%%%产生输入变量x1 x2%%%%%%%%%%%%%%%
x=rand(2,200);
%将x转换到[-1 1]之间
x=(x-0.5)*1*2;
x1=x(1,:);
x2=x(2,:);
%%%%%%%%产生输出变量y%%%%%%%%%%%%%%%%%%
y=30+x1.^2-5*cos(2*pi*x1)+3*x2.^2-5*cos(2*pi*x2);
```

```
%%%%%%%建立RBF网
络%%%%%%%%%%%%%%%%%%%
%采用approximate RBF神经网络。spread为默认值
net=newrb(x,y);
%%%%%%%建立测试样本%%%%%%%%%%%%%%%%%%
[n,m]=meshgrid(-1:0.1:1);
row=size(n);
tx1=n(:);
tx1=tx1';
tx2=m(:);
tx2=tx2';
tx=[tx1;tx2];
%%%%%%%网络仿
真%%%%%%%%%%%%%%%%%%%%%%
t=sim(net,tx);
%%%%%%%绘制3维图%%%%%%%%%%%%%%%%%%
%网络仿真得到的函数图像
p=reshape(t,row);
subplot(1,3,1)
mesh(n,m,p);
zlim([0,50])
title('仿真结果图像')
%目标函数图像
[x1, x2]=meshgrid(-1:0.1:1);
y=30+x1.^2-5*cos(2*pi*x1)+3*x2.^2-5*cos(2*pi*x2);
subplot(1,3,2)
mesh(x1,x2,y);
```

```
zlim([0,50])
title('目标函数图像')
%目标函数图象和仿真函数图象的误差图像
subplot(1,3,3)
mesh(x1,x2,y-p);
zlim([-0.1,0.05])
title('误差图像')
set(gcf,'position',[400 ,280,880,390])
```

程序仿真结果如图8-22所示。

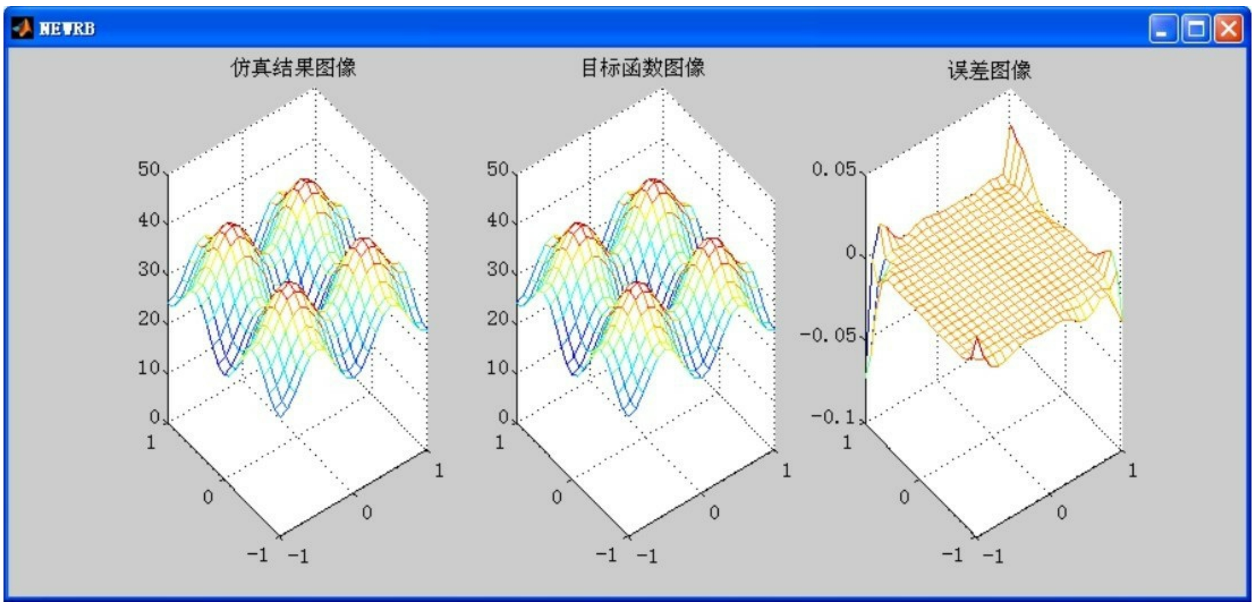


图8-22 用approximateRBF神经网络对例8-6中的函数进行拟合的结果

在软件运行过程中，MATLAB 的Command Window 会显示RBF 网络仿真输出和目标函数值之间的差值，具体如下所示。

```
NEWRB, neurons = 0, MSE = 25.5798
NEWRB, neurons = 2, MSE = 25.4481
NEWRB, neurons = 3, MSE = 25.2911
NEWRB, neurons = 4, MSE = 25.2794
```

NEWRB, neurons = 5, MSE = 23.0605

NEWRB, neurons = 6, MSE = 21.161

NEWRB, neurons = 7, MSE = 20.5395

NEWRB, neurons = 8, MSE = 17.3363

NEWRB, neurons = 9, MSE = 16.1571

NEWRB, neurons = 10, MSE = 7.67835

NEWRB, neurons = 11, MSE = 7.67831

NEWRB, neurons = 12, MSE = 7.31496

NEWRB, neurons = 13, MSE = 7.24941

NEWRB, neurons = 14, MSE = 5.97063

NEWRB, neurons = 15, MSE = 5.92575

NEWRB, neurons = 16, MSE = 5.80525

NEWRB, neurons = 17, MSE = 5.67076

NEWRB, neurons = 18, MSE = 5.46176

NEWRB, neurons = 19, MSE = 5.19379

NEWRB, neurons = 20, MSE = 4.86415

NEWRB, neurons = 21, MSE = 4.4951

NEWRB, neurons = 22, MSE = 4.14874

NEWRB, neurons = 23, MSE = 2.27968

NEWRB, neurons = 24, MSE = 2.15201

NEWRB, neurons = 25, MSE = 2.12102

NEWRB, neurons = 26, MSE = 1.79509

NEWRB, neurons = 27, MSE = 1.6103

NEWRB, neurons = 28, MSE = 1.61029

NEWRB, neurons = 29, MSE = 0.426169

NEWRB, neurons = 30, MSE = 0.292466

NEWRB, neurons = 31, MSE = 0.239106

NEWRB, neurons = 32, MSE = 0.190257

NEWRB, neurons = 33, MSE = 0.13518

NEWRB, neurons = 34, MSE = 0.134009

NEWRB, neurons = 35, MSE = 0.118296

NEWRB, neurons = 36, MSE = 0.104314

NEWRB, neurons = 37, MSE = 0.102834

NEWRB, neurons = 38, MSE = 0.10258

NEWRB, neurons = 39, MSE = 0.0957007

NEWRB, neurons = 40, MSE = 0.0826646

NEWRB, neurons = 41, MSE = 0.0708142

NEWRB, neurons = 42, MSE = 0.0511245

NEWRB, neurons = 43, MSE = 0.0309117

NEWRB, neurons = 44, MSE = 0.030777

NEWRB, neurons = 45, MSE = 0.025096

NEWRB, neurons = 46, MSE = 0.00882752

NEWRB, neurons = 47, MSE = 0.00797212

NEWRB, neurons = 48, MSE = 0.00600726

NEWRB, neurons = 49, MSE = 0.00164136

NEWRB, neurons = 50, MSE = 0.0013482

NEWRB, neurons = 51, MSE = 0.00109036

NEWRB, neurons = 52, MSE = 0.00097635

NEWRB, neurons = 53, MSE = 0.000923487

NEWRB, neurons = 54, MSE = 0.00063117

NEWRB, neurons = 55, MSE = 0.000607962

NEWRB, neurons = 56, MSE = 0.000538996

NEWRB, neurons = 57, MSE = 0.000472016

NEWRB, neurons = 58, MSE = 0.000439007

NEWRB, neurons = 59, MSE = 0.00035503

NEWRB, neurons = 60, MSE = 0.000248859

NEWRB, neurons = 61, MSE = 0.000226564

NEWRB, neurons = 62, MSE = 0.000172171

NEWRB, neurons = 63, MSE = 0.000120816

NEWRB, neurons = 64, MSE = 0.000105273

由图8-21和图8-22可知，RBF网络的仿真结果可以很好地逼近该非线性函数y，由误差图象可以看出，RBF预测效果在数据的边缘处误差较大，但对其他数据有很好的拟合效果。

RBF训练在进行到第64步的时候，RBF网络仿真输出和目标函数值之间的差值已接近0.0001，说明网络的输出能很好地逼近函数。

【例8-7】建立RBF神经网络用于图8-23所示的模式分类。

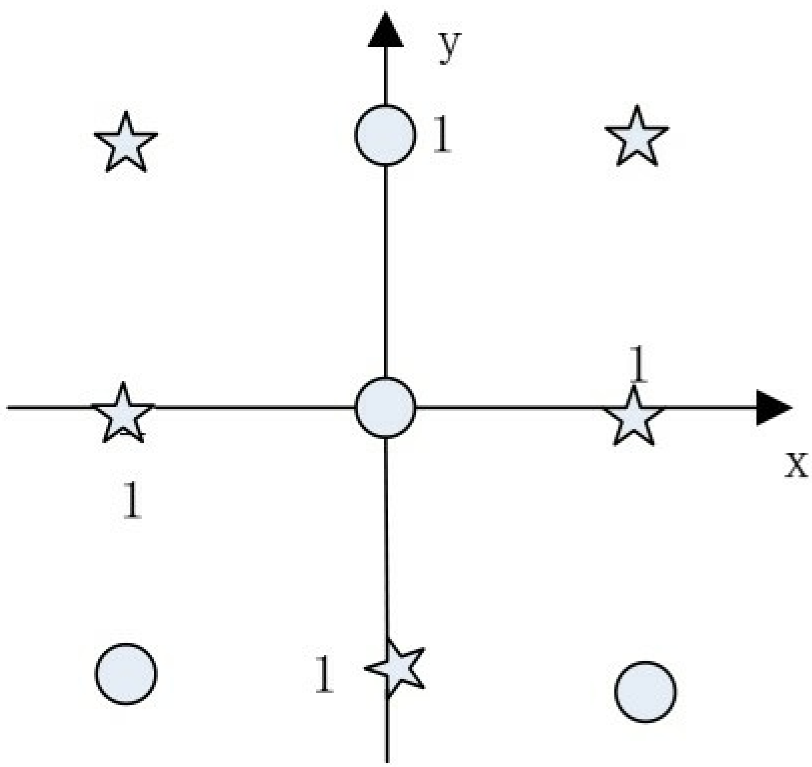


图8-23 模式分类图形

解：将圆形设为第1类模式，五角星设为第2类模式。以（x,y）代

表两种模式的位置，作为网络的输入向量。

图8-23中图形按照从左到右、从上到下的顺序排序，则RBF神经网络的输入变量如下。

```
in=[-1 0 1 -1 0 1 -1 0 1;1 1 1 0 0 0 -1 -1 -1];
```

目标变量如下。

```
t=[2 1 2 2 1 2 1 2 1];
```

根据上述输入变量和目标变量，建立RBF神经网络，MATLAB代码如下所示。

```
clear all
clc
%%%%%%%定义输入变量和目标变量%%%%%%%%%%%%
%输入变量
in=[-1 0 1 -1 0 1 -1 0 1;1 1 1 0 0 0 -1 -1 -1];
%目标变量
t=[2 1 2 2 1 2 1 2 1];
ti=ind2vec(t);
%%%%%%%设计RBF网络，并计算网络设计时间%%%%%
%计时开始
time=clock;
%建立网络
net=newpnn(in,ti,0.7);
%计算网络设计时间
datat=etime(clock,time)
%%%%%%%网络仿
真%%%%%%%%%%%%%%%%%%%%%%%%%%%
y=sim(net,in);
yv=vec2ind(y)
```

```
%%%%%%%绘制误差曲线%%%%%%%%%%%%%%%%%%%%%%
figure(1);
a=1:9;
plot(a,yv-t)
title('网络仿真误差曲线');
xlabel('输入点位置');
ylabel('仿真误差');
```

运行上述程序后，在MATLAB 的Command Window会显示设计网络所需要的时间，本例中设计网络所需要的时间为0.8280，网络仿真输出为yv。

```
datat =
   0.8280
yv =
   2   1   2   2   1   2   1   2   1
```

网络仿真误差曲线如图 8-24 所示，图中输入点位置表示将图 8-23中图形按照从左到右、从上到下的顺序排序后得到的位置，例如输入点位置为1时，表示图8-23中左上方的五角星。

由网络仿真误差曲线可以看出，网络仿真输出与目标输出完全吻合，即建立的网络很好地完成了分类。

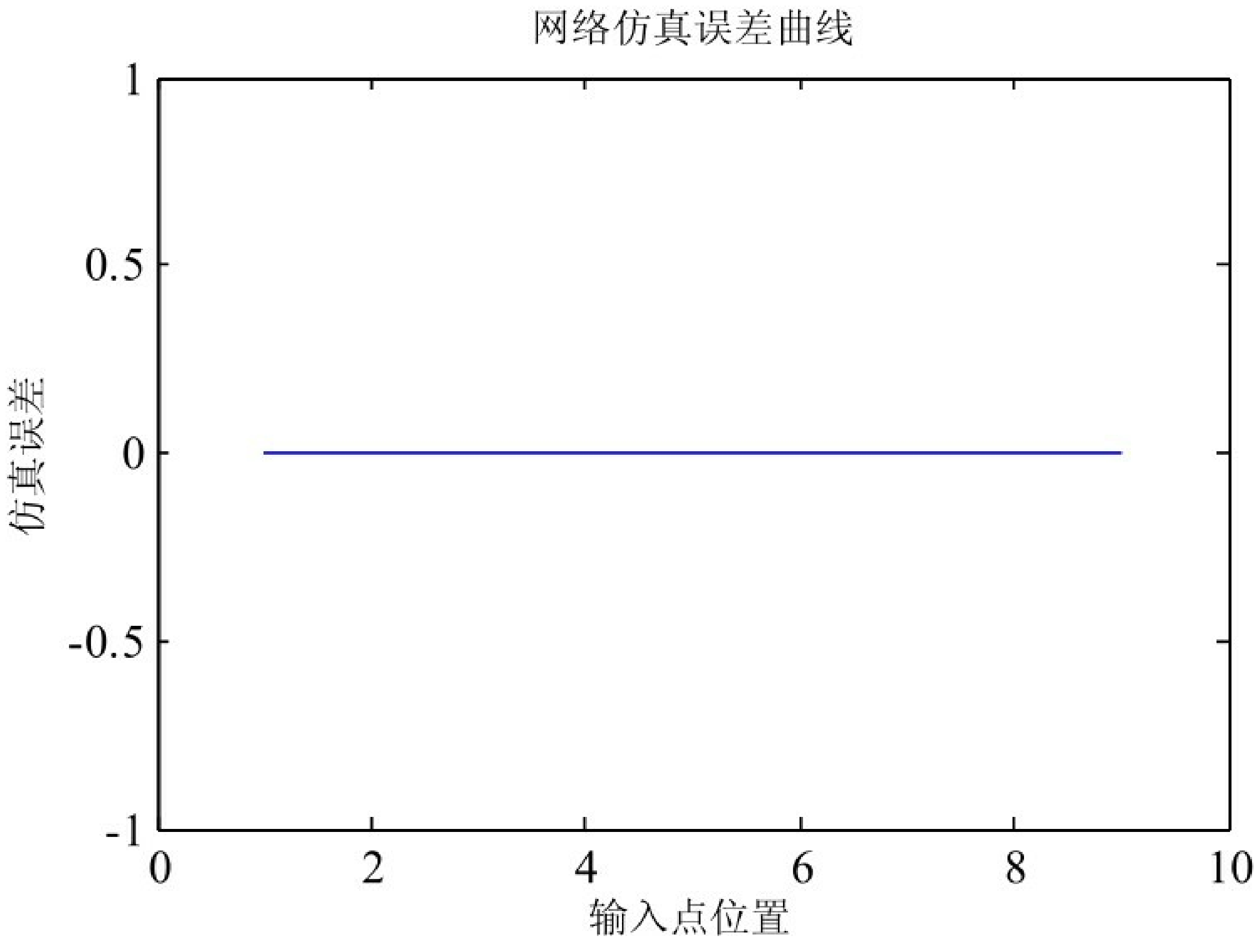


图8-24 网络仿真误差曲线

在应用RBF网络时，还需要注意，尽管RBF网络的输出是隐单元输出的线性加权和，并且网络学习速率很快，但是这并不等于RBF网络就可以取代前馈网络。

8.6 本章小结

RBF神经网络是一种高效的前馈式神经网络，它具有其他前向网络所不具有的最佳逼近性能和全局最优特性，并且结构简单，训练速度快。

本章重点介绍了RBF网络的模型及其原理、学习算法。同时对于MATLAB常用的RBF网络工具箱函数也进行了简单介绍。RBF在非线性滤波方面有很强大的优势，本章举例进行了说明。最后，通过经典案例介绍了RBF的典型应用。

第9章 反馈型神经网络

反馈网络（Recurrent Network），又称自联想记忆网络，其目的是为了设计一个网络，储存一组平衡点，使得当给网络一组初始值时，网络通过自行运行而最终收敛到这个设计的平衡点上。

反馈网络能够表现出非线性动力学系统的动态特性，它所具有的主要特性为以下两点。第一、网络系统具有若干个稳定状态，当网络从某一初始状态开始运动，网络系统总可以收敛到某一个稳定的平衡状态；第二，系统稳定的平衡状态可以通过设计网络的权值而被存储到网络中。

本章将主要介绍反馈型神经网络模型、工具箱函数及MATLAB应用实例。

学习目标：

·熟悉Hopfield 神经网络模型

·掌握Hopfield 离散型和连续型神经网络的工程应用

·掌握Elman 神经网络的工程应用

9.1 反馈型神经网络的基本概念

反馈网络包括输入延迟和输出反馈两种类型，其他的网络称为静态网络。为了对它们的不同之处有一个清晰的认识，下面举例详细说明二者的区别。

【例9-1】静态网络和反馈网络对同一个输入序列的不同响应。

解：首先设定一个输入变量，并画出其序列图。其MATLAB程序代码如下所示。

```
clear all
clc
%设定输入变量
x=[0 0 0 1 1 1 1 1 0 0 0];
%绘制输入变量序列图
cell2mat(x)
```

绘制的输入向量序列图如图9-1所示。

设定输入变量后，创建一个静态网络，并且得到这个静态网络对该输入向量序列的影响。

用下面命令创建一个单层、单神经元、没有阈值、权值为3的线性网络。

```
%创建一个单层的线性网络
net=newlin([-1 1],1);
%阈值为零
net.biasConnect=0;
%权值为3
net.iw{1,1}=3;
```

用上面创建的线性网络仿真输入变量。

```
%网络仿真
y=sim(net,x)
```

画出输出变量的序列图，如图9-2所示。

```
%绘制输出变量的序列图
stem(cell2mat(y))
```

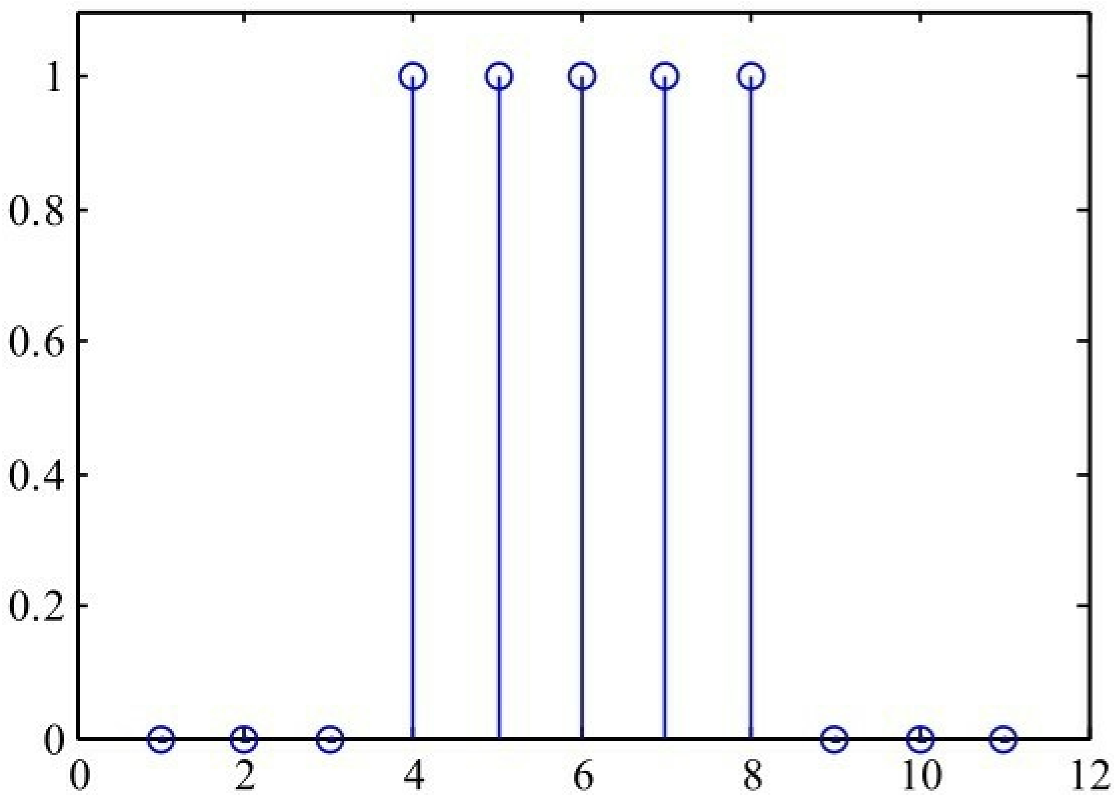


图9-1 绘制的输入变量序列图

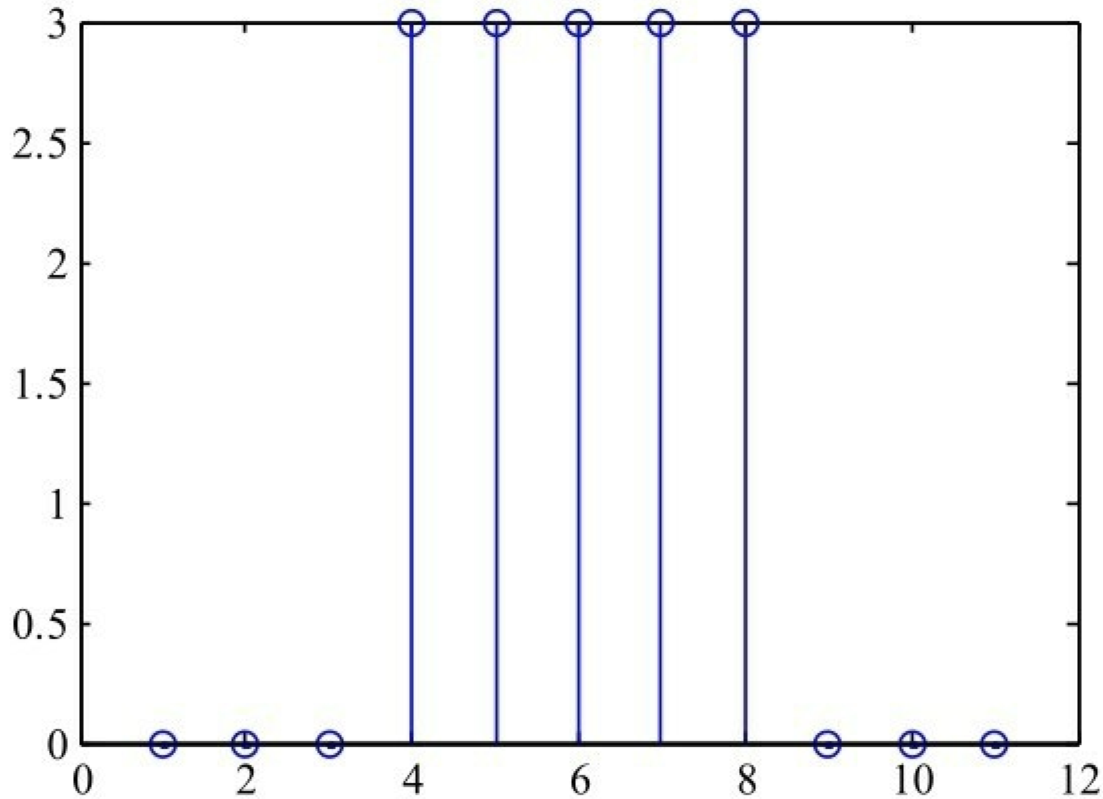


图9-2 网络仿真输出变量的序列图

图 9-2 是静态线性网络对输入序列的仿真结果。从图中可以看出，静态网络的响应所持续的时间和输入序列一样，即静态网络在任意时间点的响应只依赖于同一时间点的输入。

现在创建一个反馈神经网络。这个神经网络相当于一个输入有延迟的单层、单神经元的线性网络。创建网络及仿真、绘制输出变量的程序如下所示。

```
%创建输入有延迟的反馈网络
net=newlin([-1 1],1,[0 1]);
%偏置值为零
net.biasConnect=0;
%网络权值为[1 1]
net.iw{1,1}=[1 1];
```

%网络仿真

y=sim(net,x);

%绘制输出变量的序列图

stem(cell2mat(y))

绘制的反馈神经网络仿真得到的输出变量序列图如图9-3所示。

从图 9-3 可以看出，反馈神经网络的响应在延迟时间上要比输入序列长。因为反馈神经网络在任意时刻的响应不仅依赖于当前的输入，也依赖于以前的输入，所以它是有记忆效应的。

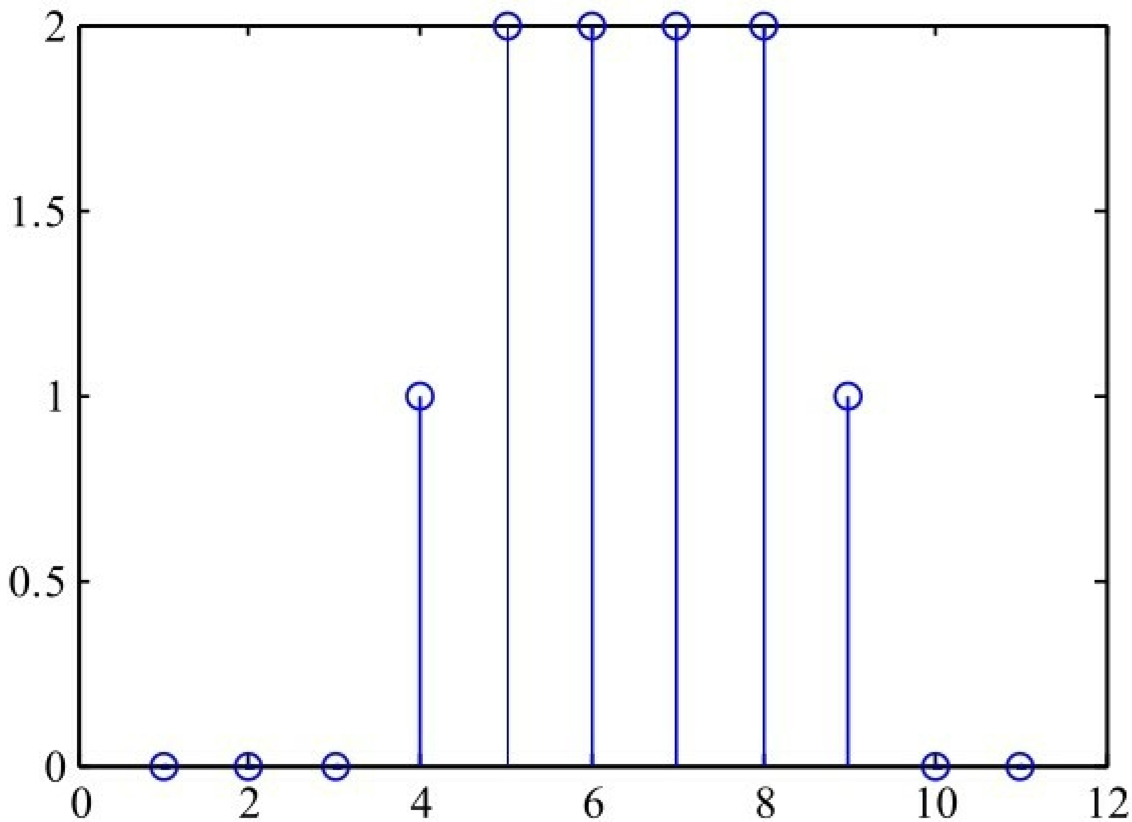


图9-3 反馈神经网络仿真得到的输出变量序列图

下面用newnarx命令创建一个输入中包括输出的反馈网络。其MATLAB代码如下所示。

%%%%%%%%%%使用newnarx命令创建网络%%%%%%%%%%

%创建单层、但神经元、输出层延迟为1的反馈网络

```
net=newnarx([-1 1],0,1,1,{'purelin'});
%阈值为0，输出层权重为0.5，输入层权重为1
net.biasConnect=0;
net.iw{1}=0.5;
net.iw{1}=1;
%对输入序列仿真
y=sim(net,x);
%绘制输出变量的序列图
stem(cell2mat(y))
```

运行程序得到图9-4所示的结果。

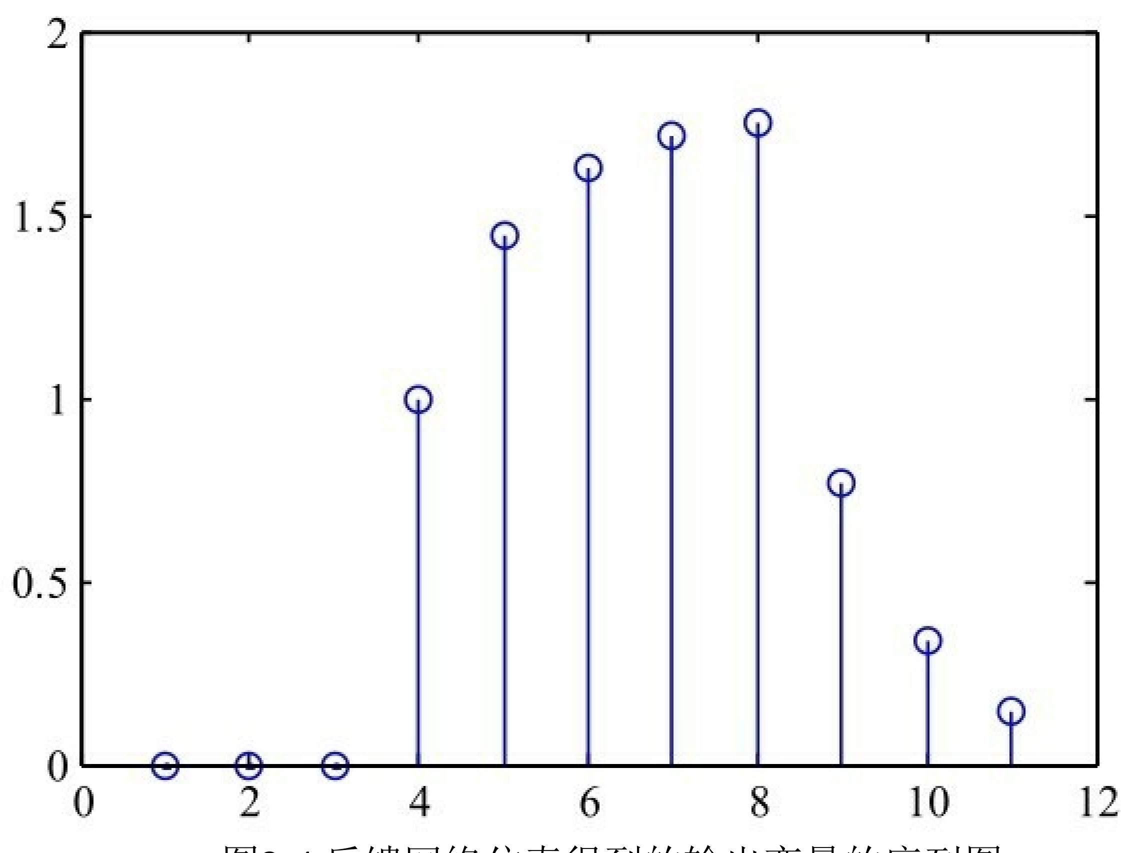


图9-4 反馈网络仿真得到的输出变量的序列图

从图9-4中可以看出，用newnarx命令创建的反馈网络比输入延迟的反馈网络有一个更长的输出响应，这是因为在时间上输出总要晚于输

入。

9.2 Hopfield网络模型

1982年，美国加州工学院物理学家霍普菲尔德（J.Hopfield）发表了一篇对人工神经网络研究颇有影响的论文。他提出了一种具有相互联结的反馈型人工神经网络模型，并将“能量函数”的概念引入到对称Hopfield网络的研究中，给出了网络的稳定性判据。

他利用多元Hopfield网络的多吸引子及其吸引域，实现了信息的联想记忆（associative memory）功能。另外Hopfield网络与电子模拟线路之间存在着明显的对应关系，使得该网络易于理解且便于实现。而它所执行的运算在本质上不同于布尔代数运算，对新一代电子神经计算机具有很大的吸引力。

反馈式网络有多种，本节主要讨论由Hopfield提出的反馈网络。

Hopfield 网络是单层对称全反馈网络，根据其激活函数的选取不同，可分为离散型的Hopfield 网络（Discrete Hopfield Neural Network，简称DHNN）和连续型的Hopfield网络（Continuous Hopfield Neural Network，简称CHNN）。

DHNN的激活函数为二值型的，其输入、输出为{0，1}的反馈网络，主要用于联想记忆。CHNN的激活函数的输入与输出之间的关系为连续可微的单调上升函数，主要用于优化计算。

Hopfield 网络已经成功地应用于多种场合，现在仍常有新的应用的报道。具体的应用方向主要集中在图像处理、语声处理、信号处理、数据查询、容错计算、模式分类、模式识别等方面。

9.2.1 Hopfield网络模型

反馈网络的网络结构如图9-5所示。

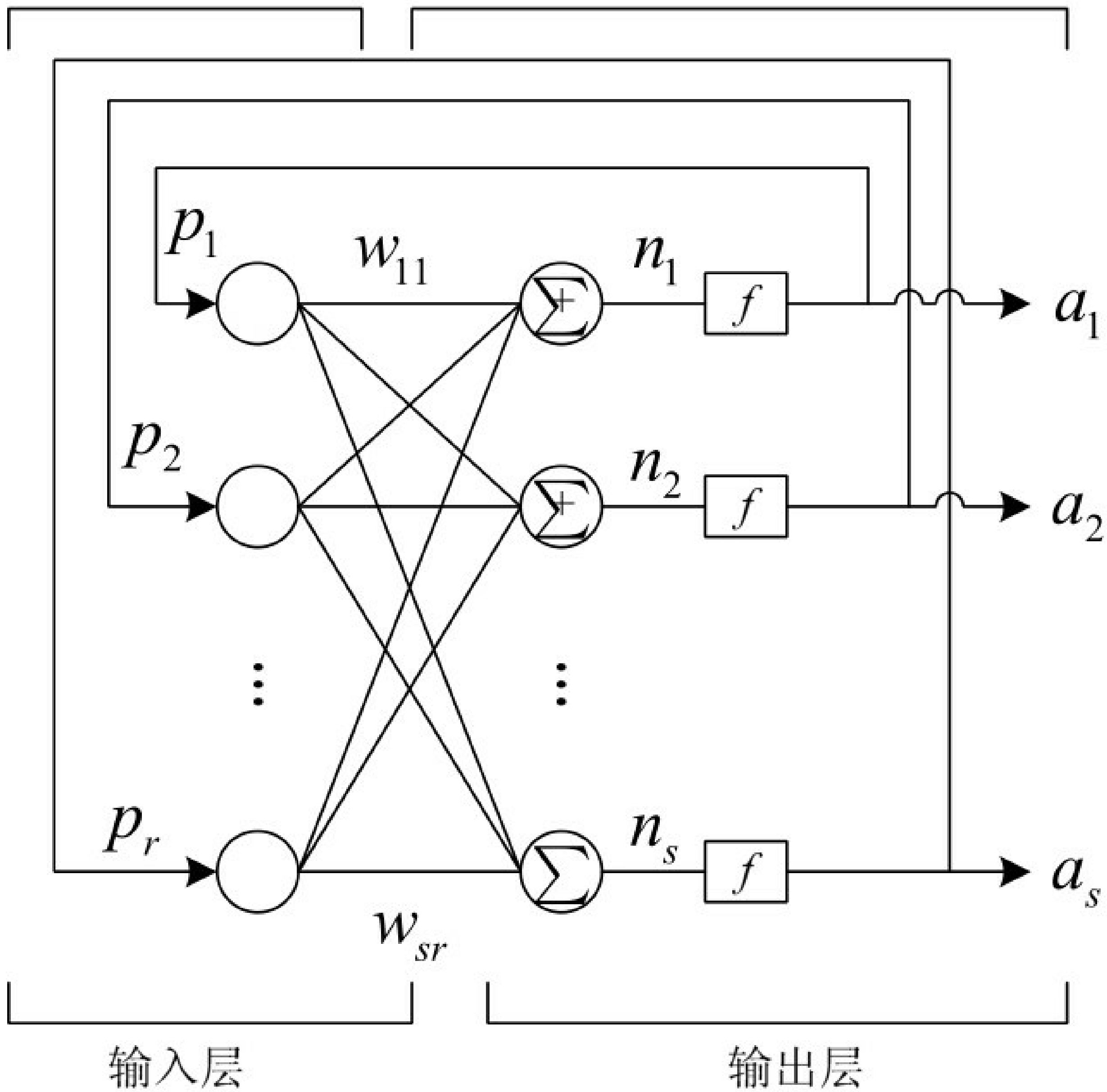


图9-5 反馈网络结构图

该网络为单层全反馈网络，其中的每个神经元的输出都是与其他神经元的输入相连的。所以其输入数目与输出层神经元的数目是相等的，有r=s。

在反馈网络中，如果其激活函数 f(·)是一个二值型的硬函数，如图9-6 所示，即ai=sgn(ni)，i=l, 2, … r，则称此网络为离散型反馈网络。

如果ai=f(ni)中的f(·)为一个连续单调上升的有界函数，这类网络被称为连续型反馈网络，图9-7所示为一个具有饱和线性的激活函数，它满足连续单调上升的有界函数的条件，常作为连续型的激活函数。

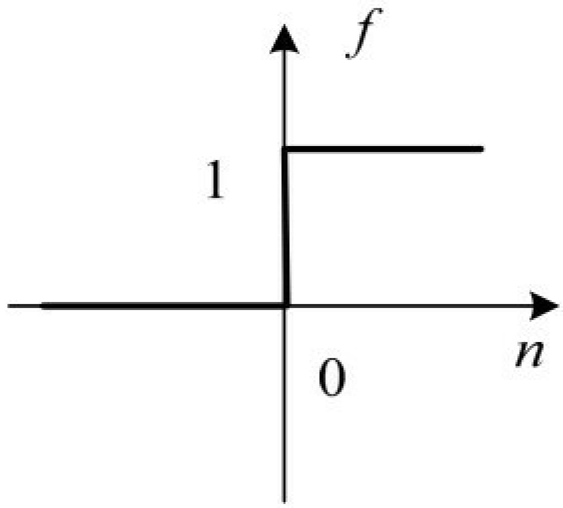


图9-6 DHNN中的激活函数

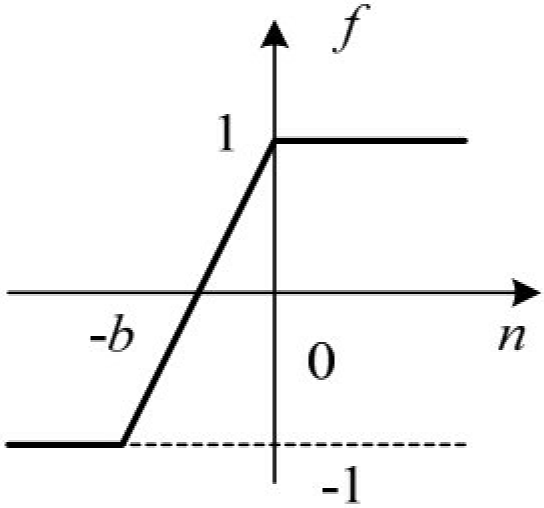


图9-7 CHNN中的激活函数

9.2.2 状态轨迹

对于一个由r个神经元组成的反馈网络，若将加权输入和n视作网络的状态，则状态矢量N=[n_1， n_2， …，n_r]，网络的输出矢量为A＝[a_1，

$a_2 \ldots$，a_s]。

在某一时刻t，分别用N(t)和A(t)来表示各自的矢量。在下一时刻t+1，可得到N(t+1)，而N(t+1)又引起A(t+1)的变化，这种反馈演化的过程，使状态矢量N(t)随时间发生变化。

在一个r维状态空间上，可以用一条轨迹来描述状态变化情况。从初始值$N(t_0)$出发，$N(t_0+\Delta t) \rightarrow N(t_0+2\Delta t) \rightarrow \ldots \rightarrow N(t_0+m\Delta t)$，这些在空间上的点组成的确定轨迹，是演化过程中所有可能状态的集合，我们称这个状态空间为相空间。

图9-8描述了一个三维相空间上3条不同的轨迹，对于DHNN，因为N(t)中每个值只可能为±1，或{0，1}，对于确定的权值w_{ij}，其轨迹是跳跃的阶梯式，如图中A所示，对于CHNN，因为f(·)是连续的，因而，其轨迹也是连续的。如图中B、C所示。

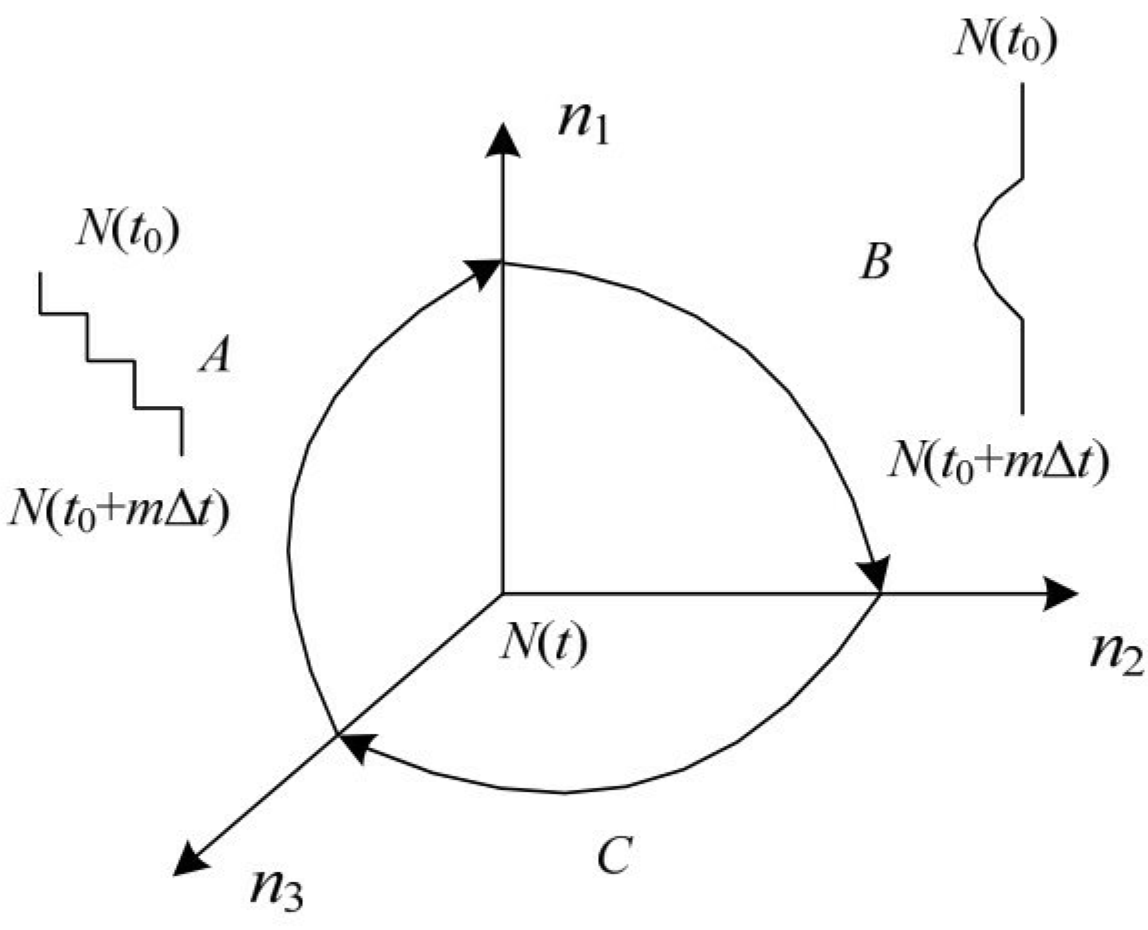


图9-8 三维空间中的状态轨迹

对于不同的连接权值w_{ij}和输入P_j(i, j=1, 2, … r)，反馈网络状态轨迹可能出现不同的情况。

9.2.3 状态轨迹发散

如果状态 N(t)的轨迹随时间一直延伸到无穷远，此时状态发散，系统的输出也发散。在人工神经网络中，由于输入、输出激活函数上一个有界函数，虽然状态 N(t)是发散的，但其输出A(t)还是稳定的，而A(t)的稳定反过来又限制了状态的发散。

一般非线性人工神经网络中发散现象是不会发生的，除非神经元的输入输出激活函数是线性的。

对于一个由 r 个神经元组成的反馈系统，它的行为就是由这些状态轨迹的情况来决定的。目前的人工神经网络是利用第一种情况即稳定的专门轨迹来解决某些问题的。

如果把系统的稳定点视作一个记忆的话，那么从初始状态朝这个稳定点移动的过程就是寻找该记忆的过程。状态的初始值可以认为是给定的有关该记忆的部分信息，状态 N(t)移动的过程，是从部分信息去寻找全部信息，这就是联想记忆的过程。

如果把系统的稳定点考虑为一个能量函数的极小点，在状态空间中，从初始状态$N(t_0)$＝$N(t_0+t)$，最后到达N。若N为稳定点，则可以看作是N把$N(t_0)$吸引了过去，在$N(t_0)$时能量比较大，而吸引到N时能量已为极小。

根据这个道理，可以把这个能量的极小点作为一个优化目标函数的极小点，把状态变化的过程看成是优化某一个目标函数的过程。

因此反馈网络的状态移动的过程实际上是一种计算联想记忆或优化的过程。它的解并不需要真的去计算，只需要去形成一类反馈神经网

络，适当地讨论其权重值w_{ij}，使其初始输入$A(t_0)$向稳定吸引子状态的移动就可以达到这个目的。

Hopfield 网络是利用稳定吸引子来对信息进行储存的，利用从初始状态到稳定吸引子的运行过程来实现对信息的联想存取的。通过对神经元之间的权和阈值的设计，要求单层的反馈网络达到下列目标。

（1）网络系统能够达到稳定收敛。

即研究系统在什么条件下不会出现振荡和混沌现象。

（2）网络的稳定点。

一个非线性网络能够有很多个稳定点，对权值的设计，要求其中的某些稳定点是所要求的解。对于用作联想记忆的反馈型网络，希望稳定点就是一个记忆，那么记忆容量就与稳定点的数量有关，希望记忆的量越大，那么，稳定点的数目也越大，但稳定点数目的增加可能会引起吸引域的减小，从而使联想功能减弱。

对于用作优化的反馈网络，由于目标函数（即系统中的能量函数）往往要求只有一个全局最小。那么稳定点越多，陷入局部最小的可能性就越大，因而要求系统的稳定点越少越好。

（3）吸引域的设计。

希望的稳定点有尽可能大的吸引域，而非希望的稳定点的吸引域要尽可能的小。因为状态空间是一个多维空间，状态随时间的变化轨迹可能是多种形状，吸引域就很难用一个明确的解析式来表达，这在设计时要尽可能考虑。

9.3 Hopfield网络工具箱函数

MATLAB神经网络工具箱包含了很多用于Hopfield网络设计与分析的函数，本节重点介绍几个常用函数的功能、调用格式和参数含义等。

9.3.1 Hopfield网络创建函数

Hopfield 网络的创建函数是newhop( )，该函数的功能是生成一个Hopfield 回归网络。其调用格式如下。

net = newhop(T)

其中，net 为生成的神经网络，且在T 中有稳定的点；T 是具有Q 个目标向量的R·Q矩阵（元素必须为−1或1）。

【例9-2】利用newhop函数，设定有2列的目标向量，建立具有2个稳定点的Hopfield 神经网络。

解：首先设定具有2列目标向量的矩阵。

```
T = [+1 -1;-1 +1];
```

绘制两个稳定点的Hopfield神经网络稳定空间图形，如图9-9所示。

```
plot(T(1,:),T(2,:),'r*')
axis([-1.1 1.1 -1.1 1.1])
title('Hopfield Network State Space')
xlabel('a(1)');
ylabel('a(2)');
```

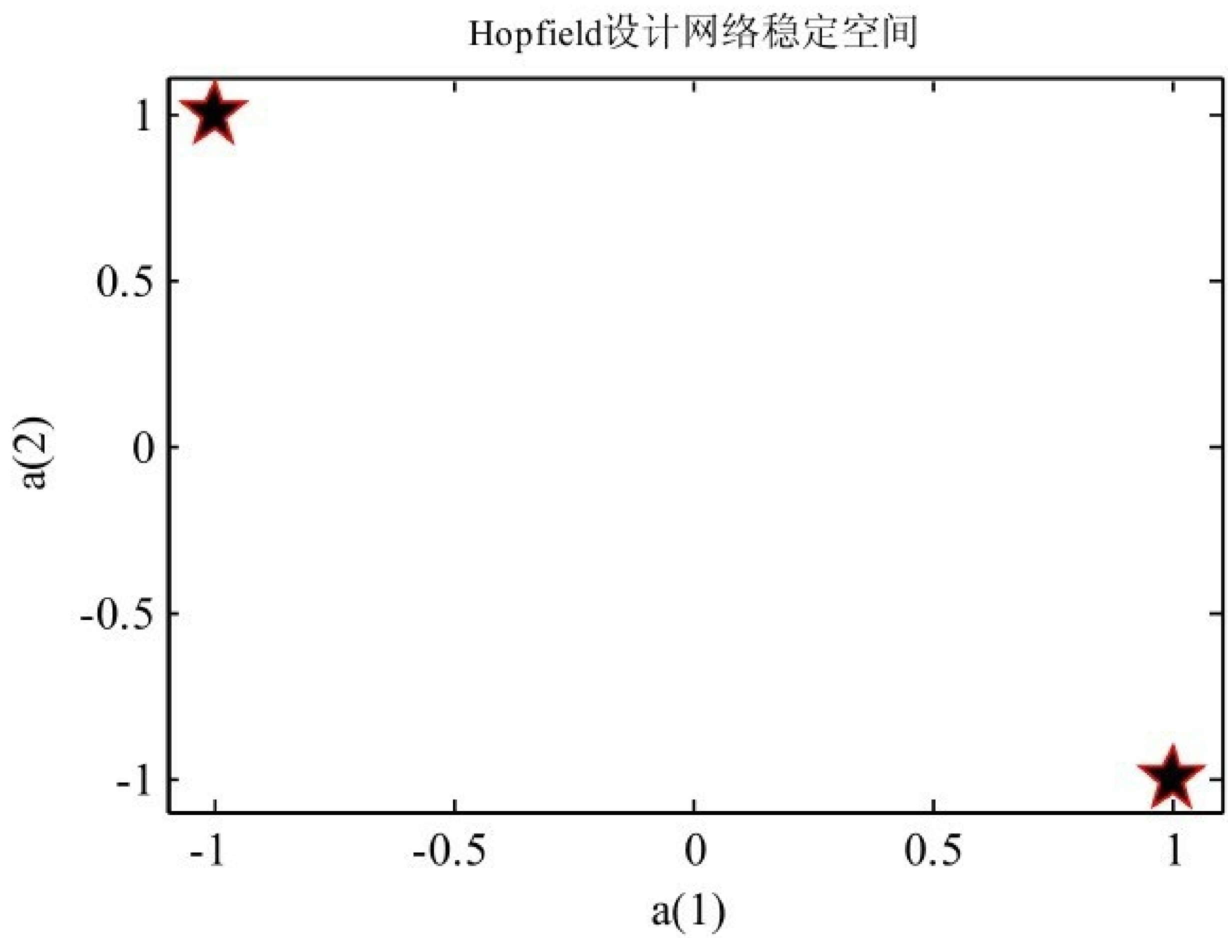


图9-9 Hopfield神经网络稳定空间图形

Hopfield网络的所有可能的状态都包含在图中边界内。

```
%使用newhop创建Hopfield神经网络
net = newhop(T);
%确定目标向量处于稳定
[Y,Pf,Af] = net([],[],T);
Y
```

运行上面代码后，在MATLAB 的Command Window窗口得到如下结果。

```
>> demohop1
Y =
   1     -1
```

-1   1

由上可知，网络返回的两个目标与目标向量T一致，所以建立的Hopfield网络处于稳定状态。

```
%定义随机的起始点
a ={rands(2,1)};
%Hopfield仿真参数设定
[y,Pf,Af] = net({20},{},a);
%在网络中设定一个活动点，确定网络的两个目标点位置是否不变。
record = [cell2mat(a) cell2mat(y)];
start = cell2mat(a);
hold on
plot(start(1,1),start(2,1),'bx',record(1,:),record(2,:))
```

运行以上代码得到图9-10所示图形，由图可以看出，Hopfield网络的两个目标点位置没变，即在左上方和右下方。

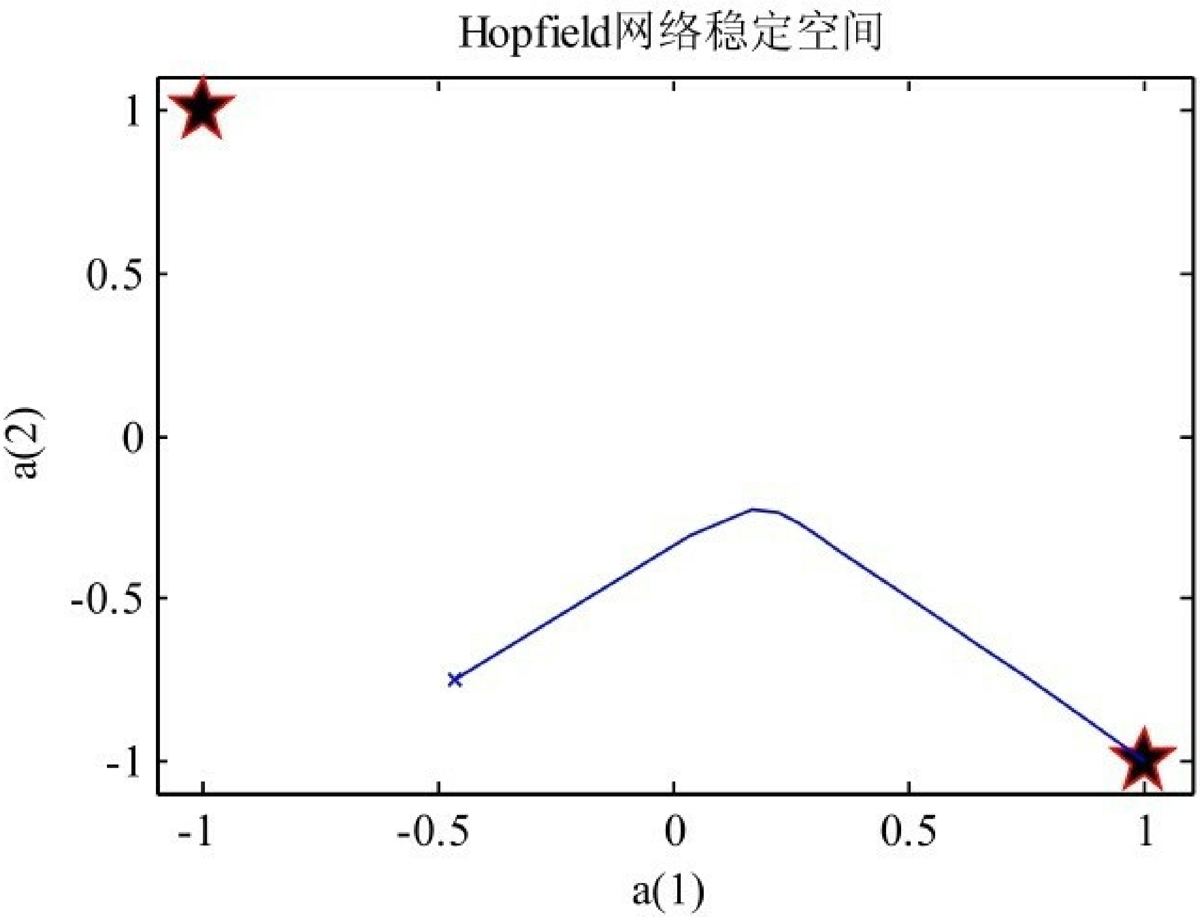


图9-10 有一个活动点的稳定空间

```
%重复模拟25个初始条件
color = 'rgbmy';
for i=1:25
    a ={rands(2,1)};
    [y,Pf,Af] = net({20},{},a);
    record=[cell2mat(a) cell2mat(y)];
    start=cell2mat(a);
    plot(start(1,1),start(2,1),'kx',record(1,:),record(2,:),color(rem(i,5)+1))
end
```

重复模拟25个初始条件得到的图形如图9-11所示。

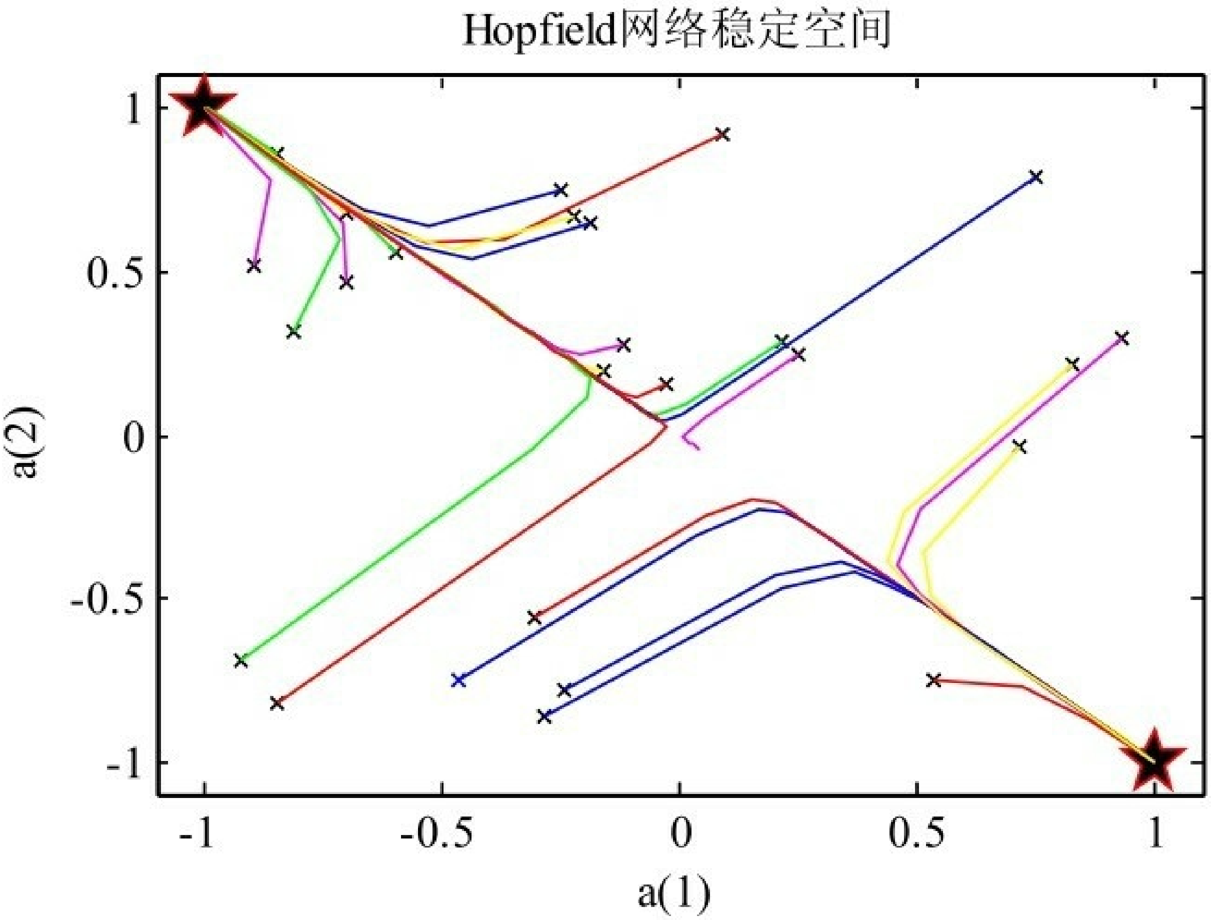


图9-11 有25个活动点的稳定空间

在图9-11中，如果Hopfield网络开始接近左上角，它会去左上角，反之亦然。这种找到初始输入最近的记忆是Hopfield神经网络的典型能力。

Hopfield神经网络经常被应用于模式的联想记忆中。Hopfield神经网络仅有一层，其激活函数用satlins(　)函数，层中的神经元有来自它自身的连接权和阈值。

【例9-3】利用newhop函数，设定有3列的目标向量，建立具有2个稳定点的Hopfield 神经网络。

解：建立题中所要求的网络，MATLAB代码如下所示。

```
% 定义具有3 列的目标向量
T = [+1 +1;-1 +1;-1 -1];
```

```
%绘制两个稳定点的Hopfield神经网络稳定空间图形
axis([-1 1 -1 1 -1 1])
set(gca,'box','on'); axis manual; hold on;
plot3(T(1,:),T(2,:),T(3,:),'r*')
title('Hopfield Network State Space')
xlabel('a(1)');
ylabel('a(2)');
zlabel('a(3)');
view([37.5 30]);
```

两个稳定点的Hopfield神经网络稳定空间图形如图9-12所示。

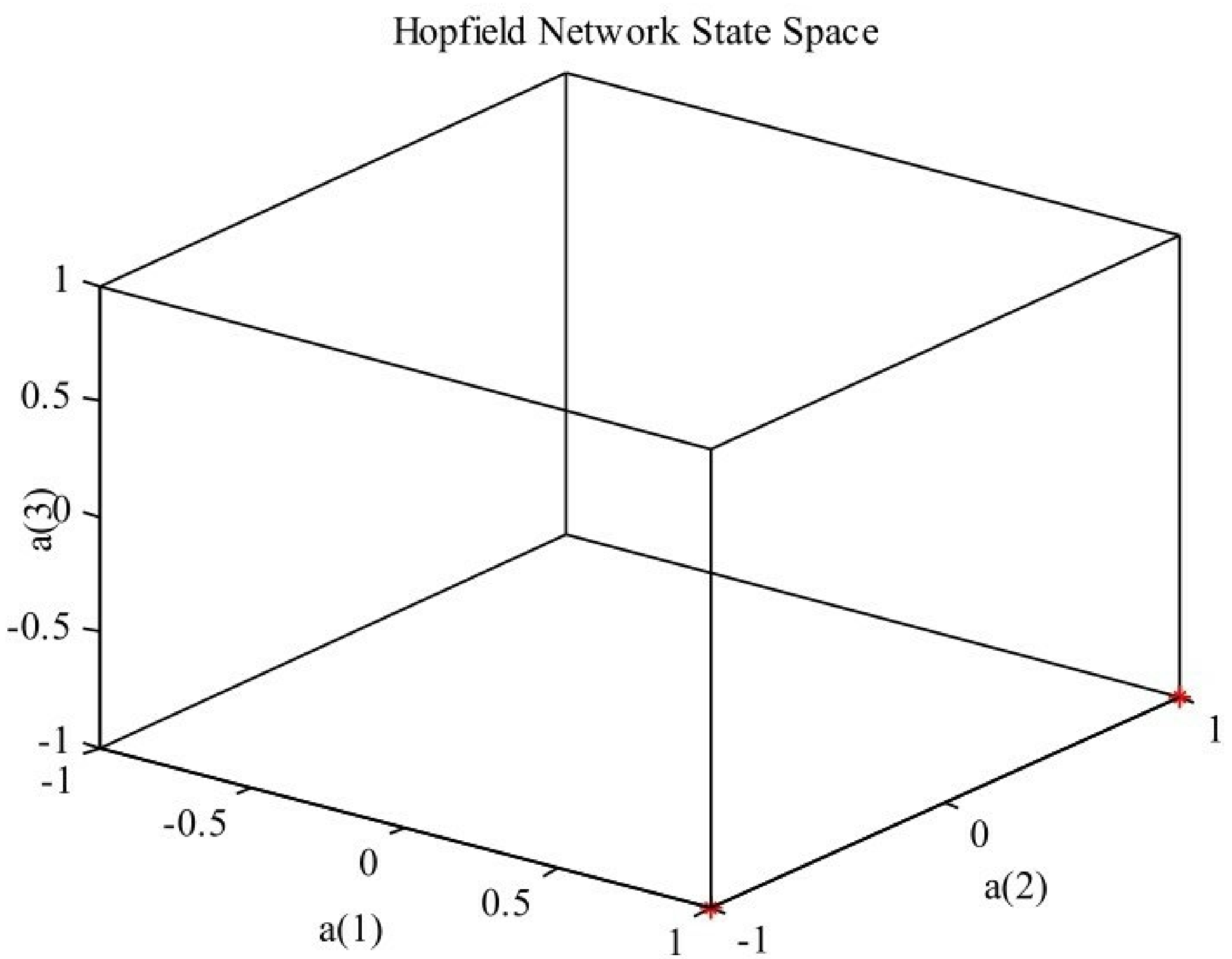


图9-12 Hopfield神经网络稳定空间图形

```
%利用newhop创建Hopfield神经网络
net = newhop(T);
% 定义随机起始点
a ={rands(3,1)};
%Hopfield仿真参数设定
[y,Pf,Af] = net({1 10},{},a);
%在稳定空间内设定一个活动的点
record = [cell2mat(a) cell2mat(y)];
start = cell2mat(a);
hold on
plot3(start(1,1),start(2,1),start(3,1),'bx', ...
    record(1,:),record(2,:),record(3,:))
```

在稳定空间内设定一个活动的点的图形如图9-13所示。

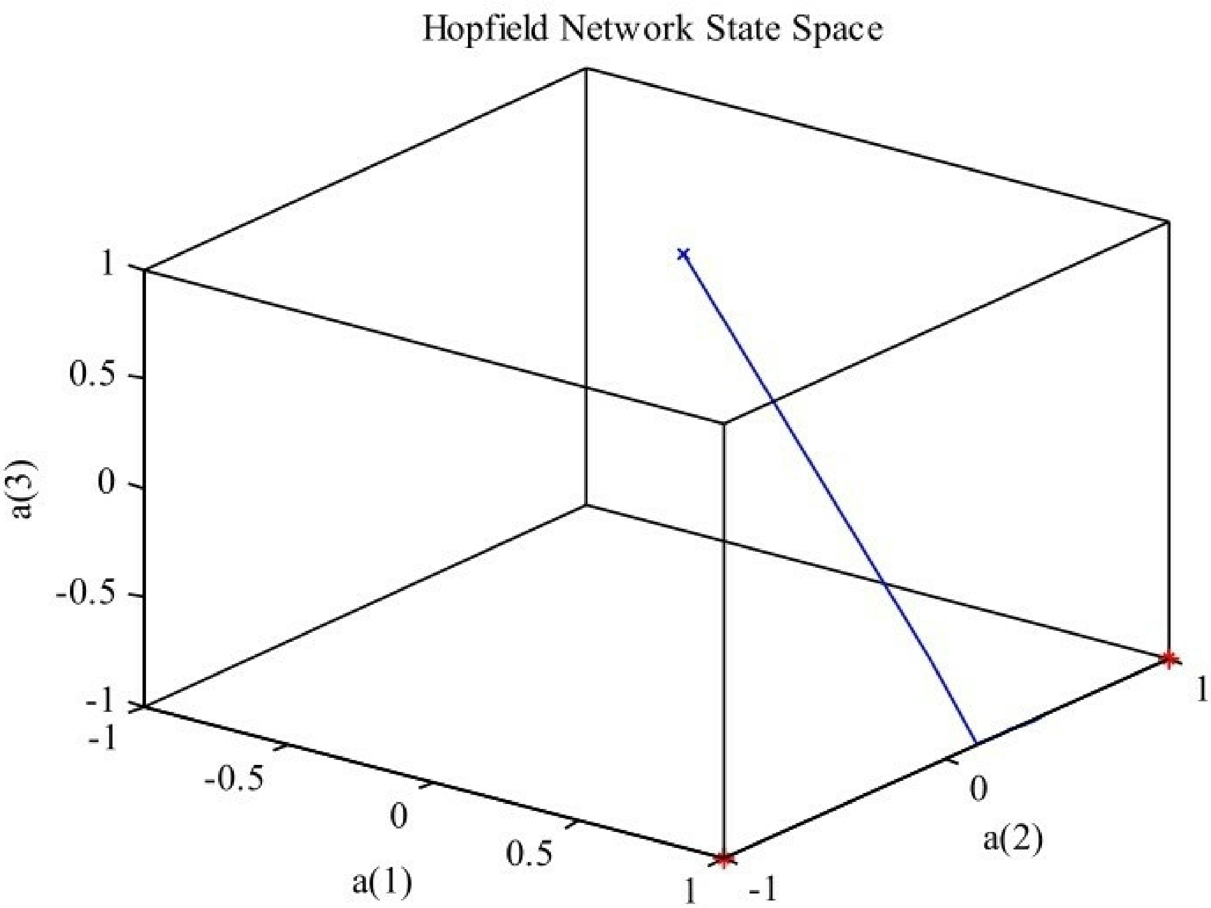


图9-13 在稳定空间内设定一个活动的点

```
%重复模拟25个初始条件
color = 'rgbmy';
for i=1:25
    a ={rands(3,1)};
    [y,Pf,Af] = net({1 10},{},a);
    record=[cell2mat(a) cell2mat(y)];
    start=cell2mat(a);
    plot3(start(1,1),start(2,1),start(3,1),'kx', ...
        record(1,:),record(2,:),record(3,:),color(rem(i,5)+1))
end
```

重复模拟25个起始点得到的稳定空间如图9-14所示。

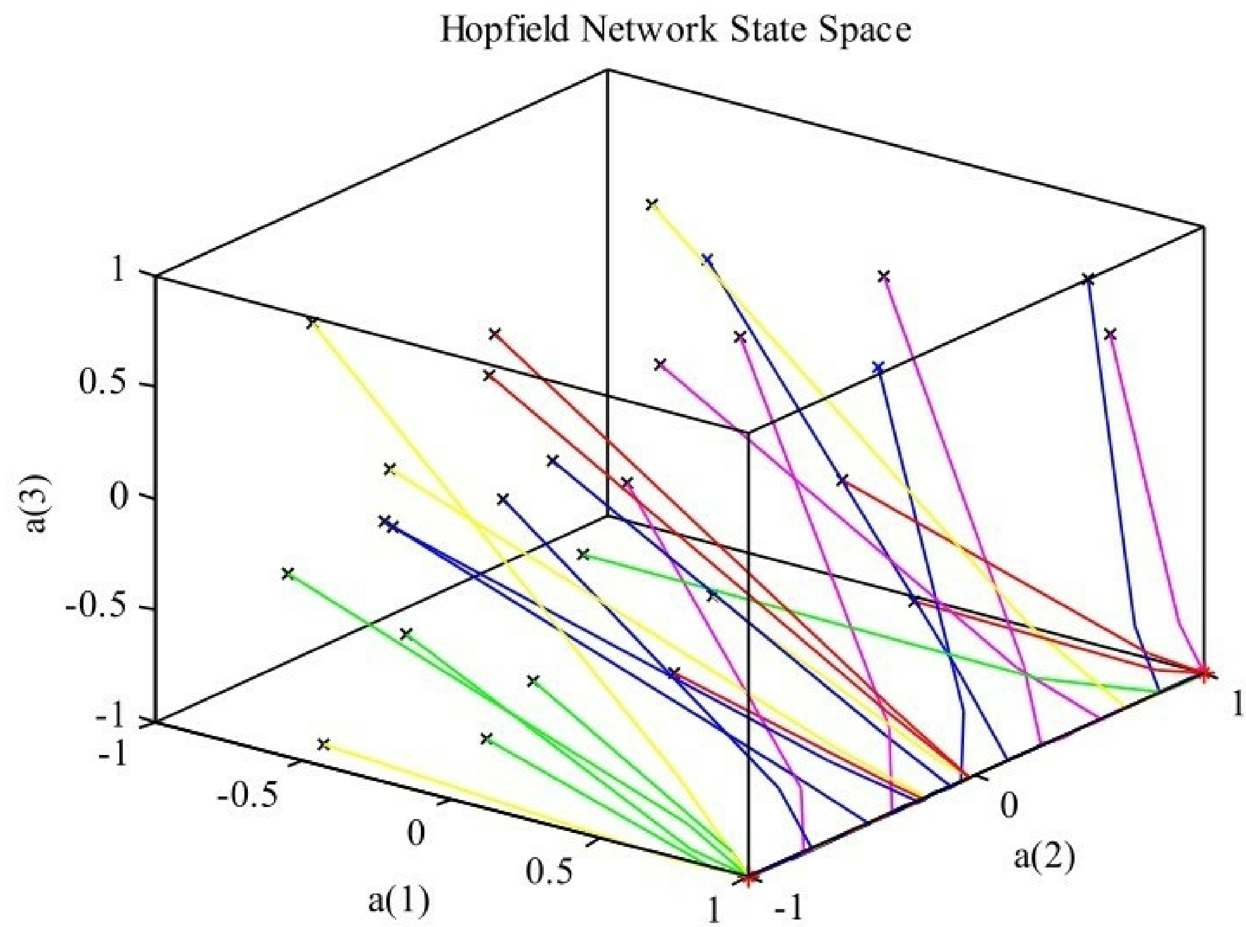


图9-14 重复模拟25个起始点得到的稳定空间

%使用向量P的每一列仿真Hopfield神经网络

P= [1.0 -1.0 -0.5 1.00 1.00 0.0; ...0.0 0.0 0.0 0.00 0.00 -0.0; ...-1.0 1.0 0.5 -1.01 -1.00 0.0];

cla

plot3(T(1,:),T(2,:),T(3,:),'r*')

color = 'rgbmy';

for i=1:6

  a ={P(:,i)};

  [y,Pf,Af] = net({1 10},{},a);

  record=[cell2mat(a) cell2mat(y)];

  start=cell2mat(a);

```
  plot3(start(1,1),start(2,1),start(3,1),'kx', ...
      record(1,:),record(2,:),record(3,:),color(rem(i,5)+1))
end
```

上述代码使得两个目标稳定点之间的起始点都进入稳定空间的中心，示意图如图9-15所示。

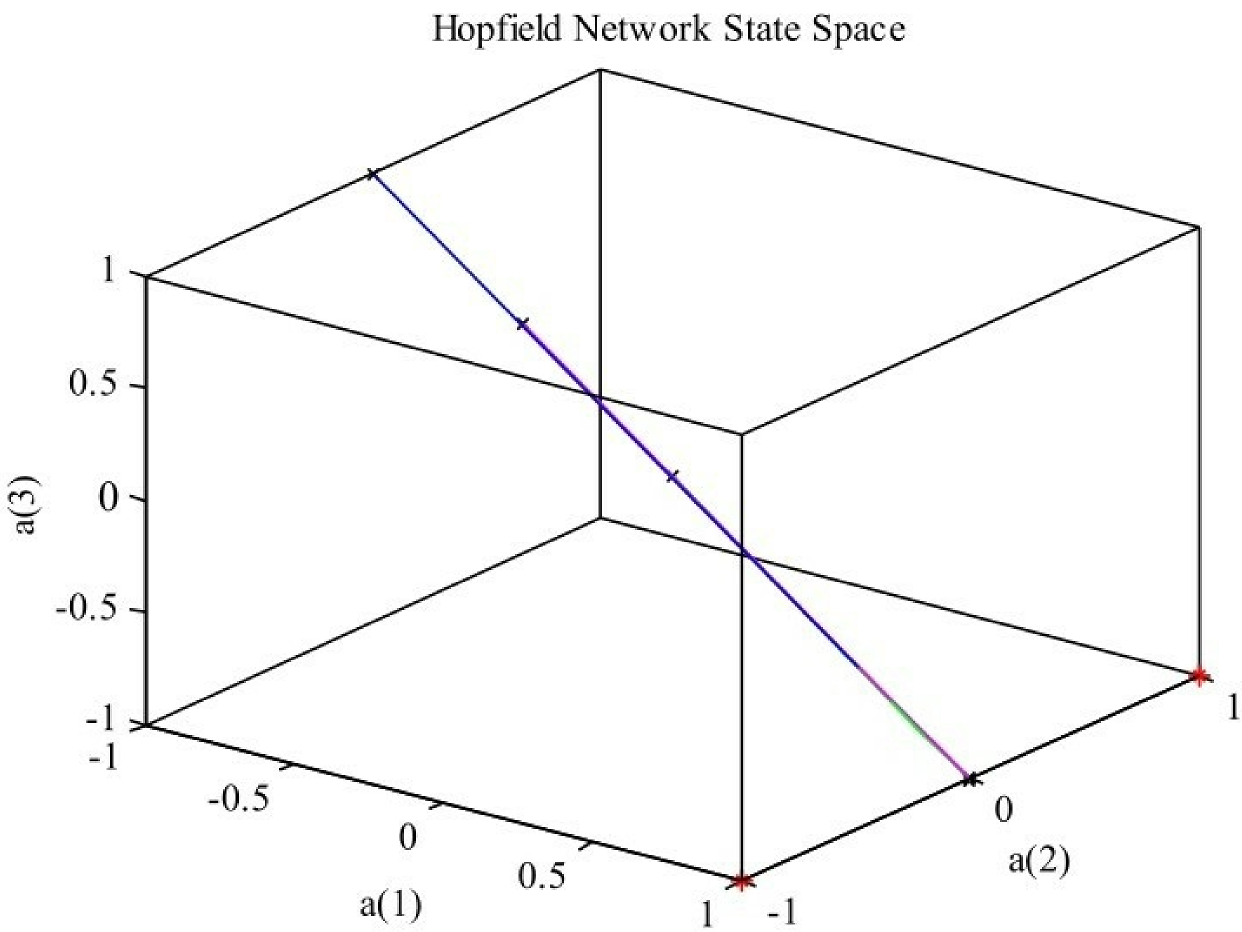


图9-15 将两个目标稳定点之间的起始点都进入稳定空间中心示意图

9.3.2 Hopfield网络传递函数

1.satlin( )

该函数的功能是饱和线性传递函数，其调用格式如下。

A = satlin(N)

其中，A 是输出向量矩阵；N 是由网络的输入向量组成的S·Q 矩

阵，返回的矩阵A　与N的维数大小一致，A的元素取值位于区间[0，1]内。当N中的元素介于0和1之间时，其输出等于输入；当输入值小于0时返回−1；当输入值大于1时返回1。

通过下面MATLAB程序得到传递函数satlin的图形如图9-16所示。

```
n =-5:0.1:5;
a = satlin(n);
plot(n,a)
```

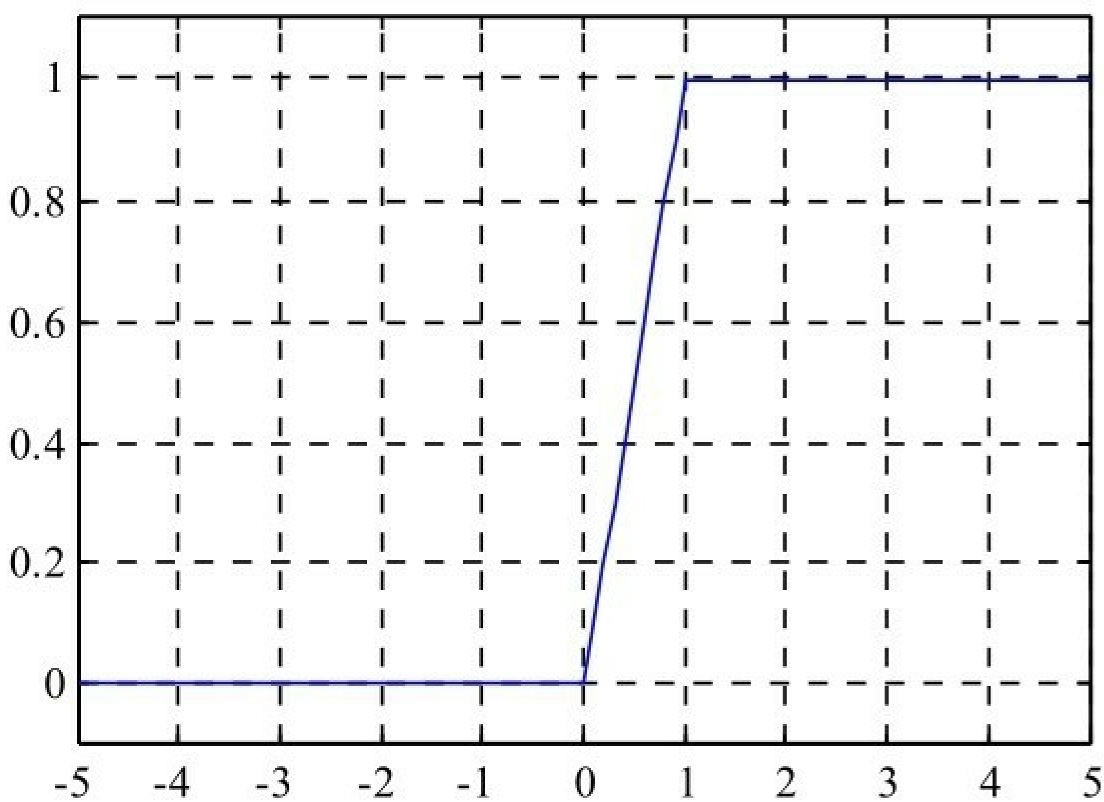


图9-16 饱和线性传递函数图形

2.satlins( )

该函数的功能是对称饱和线性传递函数，其调用格式如下。

A = satlins(N)

其中，A 输出向量矩阵；N 是由网络的输入向量组成的S·Q 矩阵，返回的矩阵A　与N的维数大小一致，A的元素取值位于区间[-1，1]内。当N中的元素介于−1和1之间时，其输出等于输入；当输入值小于−1时

返回−1；当输入值大于1时返回1。

通过下面MATLAB程序得到传递函数satlins的图形如图9-17所示。

```
n =-5:0.1:5;
a = satlins(n);
plot(n,a)
```

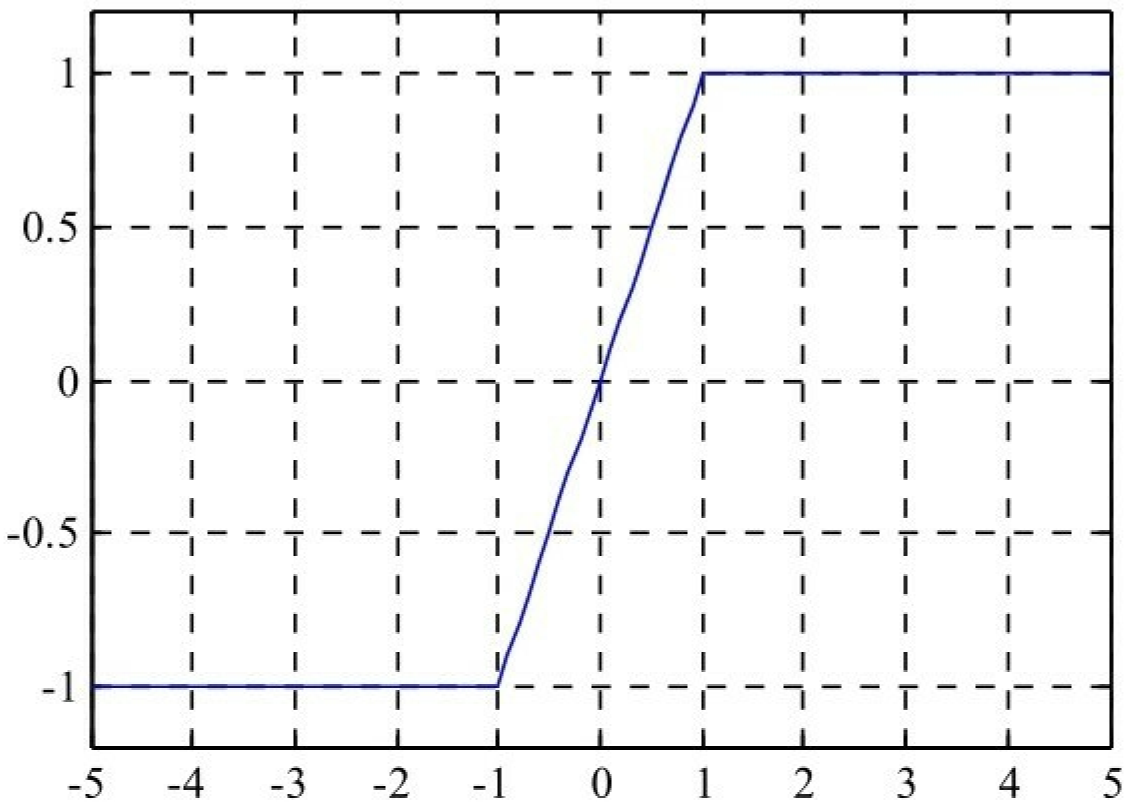


图9-17 对称饱和线性传递函数图形

9.4 离散型Hopfield网络

9.4.1 DHNN模型结构

在DHNN模型中，每个神经元节点的输出可以有两值状态：−1或1(0或1)，其输出类似于MP神经元。假设DHNN有n个神经元，每个神经元的输出为a ，权值为w，偏置为b，神经元学习函数为f，则第i+1个神经元的输出a_{i+1}可表示为：

$$a_{i+1}=\begin{cases}1 & f(a_i)=\sum_{j=1,i=1}^{n}(w_{ij}a_i+b_i)\geq 0\\ -1 & f(a_i)=\sum_{j=1,i=1}^{n}(w_{ij}a_i+b_i)<0\end{cases}$$

在上式中，取$b_i=0$，权矩阵中有$w_{ij}=w_{ji}$，且取$w_{ii}=0$，即DHNN采用对称联接。因此，其网络结构可以用一个加权元向量图表示。图9-18（a）所示为一个3节点DHNN结构，其中，每个输入神经元节点除了不与具有相同节点号的输出相连外，与其他节点两两相连。每个输出信号又反馈到相同的输入节点。

由图9-18（a），考虑到DHNN的权值特性$w_{ij}=w_{ji}$，网络各节点加权输入和分别为

$$\begin{cases} s_1 = w_{12}a_2 + w_{13}a_3 \\ s_2 = w_{21}a_1 + w_{23}a_3 \\ s_3 = w_{31}a_1 + w_{13}a_2 \end{cases}$$

由此可得简化后等效的网络结构如图9-18（b）所示。

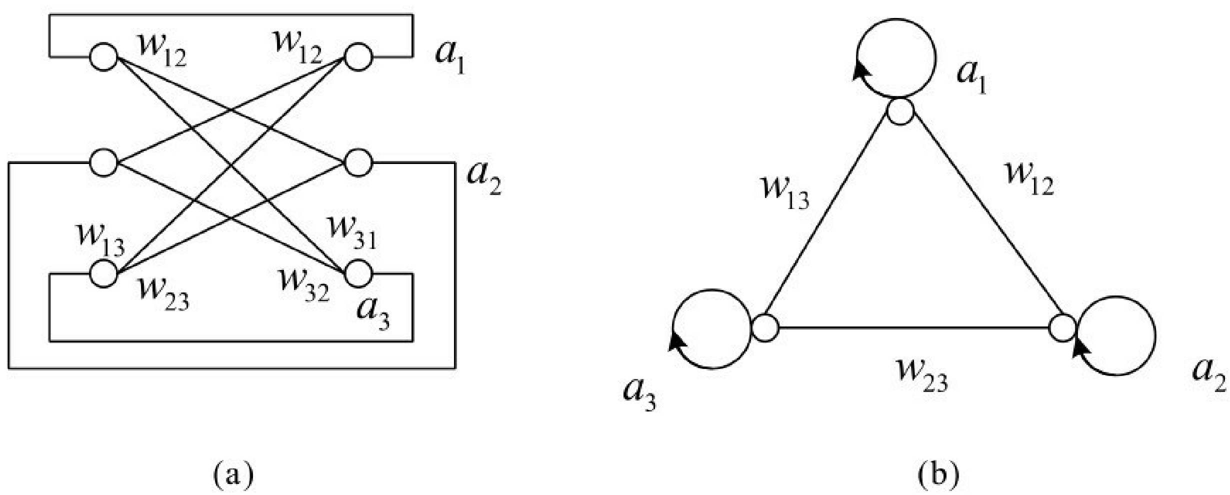


图9-18 Hopfield网络图

对于以符号函数为激活函数的网络，网络的方程可写为

$$\begin{cases} n_j(t) = \sum_{\substack{i=1 \\ i \neq j}}^{r} w_{ij} a_j \\ a_j(t+1) = \mathrm{sgn}[n_j(t)] \\ \qquad = \begin{cases} 1 & n_j(t) \geq 0 \\ -1 & n_j(t) < 0 \end{cases} \end{cases}$$

9.4.2 联想记忆

联想记忆功能是DHNN的一个重要应用范围。要实现联想记忆，反馈网络必须具有两个基本条件。一是网络能收敛到稳定的平衡状态，并以其作为样本的记忆信息；二是具有回忆能力，能够从某一残缺的信息回忆起所属的完整的记忆信息。

DHNN实现联想记忆的过程分为两个阶段：学习记忆阶段和联想回忆阶段。

在学习记忆阶段中，设计者通过某一设计方法确定一组合适的权值，使网络记忆期望的稳定平衡点。而联想回忆阶段则是网络的工作过程。此时，给定网络某一输入模式，网络能够通过自身的动力学状态演化过程最终达到稳定的平衡点，从而实现自联想或异联想回忆。

反馈网络有两种基本的工作方式：串行异步和并行同步方式。

（1）串行异步方式：任意时刻随机地或确定性地选择网络中的一个神经元进行状态更新，而其余神经元的状态保持不变。

（2）并行同步方式：任意时刻网络中部分神经元（比如同一层的神经元）的状态同时更新。如果任意时刻网络中全部神经元同时进行状态更新，那么称之为全并行同步方式。

对于s个神经元的反馈网络DHNN有2s个状态的可能性。其输出状态是一个包含−1或1（0或1）的矢量，每一时刻网络将处于某一种状态下。当网络状态的更新变化采用随机异步策略，即随机地选择下一个要更新的神经元时每个神经元将具有相同的变化概率。

在状态更新过程中，包括3种情况：由−1变为1；由1变为−1及状态保持不变。在任一时刻，网络中只有一个神经元被选择进行状态更新或保持，所以异步状态更新的网络从某一初态开始需经过多次更新状态后才可以达到某种稳态。

这种更新方式的特点是，实现上容易，每个神经元有自己的状态更

新时刻，不需要同步机制；另外，功能上的串行状态更新可以限制网络的输出状态，避免不同稳态等概率的出现；再者，异步状态更新更接近实际的生物神经系统的表现。

在日常生活中，经常会遇到带噪声字符的识别问题，如交通系统中的汽车牌照。由于汽车在使用过程中经受的风吹日晒，其牌照上的字就会模糊不清，难以辨认。如何从这些残缺不全的字符中获取完整且正确的信息，是字符识别的关键。

目前有很多种方法用于字符识别，主要分为神经网络识别、概率统计识别和模糊识别等。传统的数字识别方法在有干扰的情况下，不能很好地对数字进行识别。但是离散型Hopfield神经网络具有联想记忆功能，利用这一功能，可以有效地进行数字识别。

下面举例说明Hopfield神经网络在识别中的应用。

【例 9-4】设印刷体数字由 10×10 点阵构成，就是将数字分成很多小方块，每个方块对应数字的一部分，构成数字本部分的方块用1表示，空白处用−1表示。试设计一个Hopfield网络，能够正确识别模糊的印刷体的数字。

解：假定网络由0～9共10个稳态构成，而每个稳态用10×10的矩阵表示。这个矩阵可以直观描述阿拉伯数字，即将数字划分为10×10的矩阵，有数字的部分用1表示，空白的部分用−1表示。

网络对这10个稳态具有联想记忆的功能，当带有噪声的数字点阵输入到该网络时，网络的输出便可以得到最接近的目标向量，从而达到正确识别的效果。

Hopfield神经网络的设计步骤如图9-19所示。



图9-19 Hopfield神经网络设计步骤

因为每个数字的识别方法基本一样，所以这里只讲述数字1的识别

方法。根据识别数字1的方法，数字0,2,3，…，9等9个数字也可以进行识别。

假设数字1的点阵组成如图9-20所示。

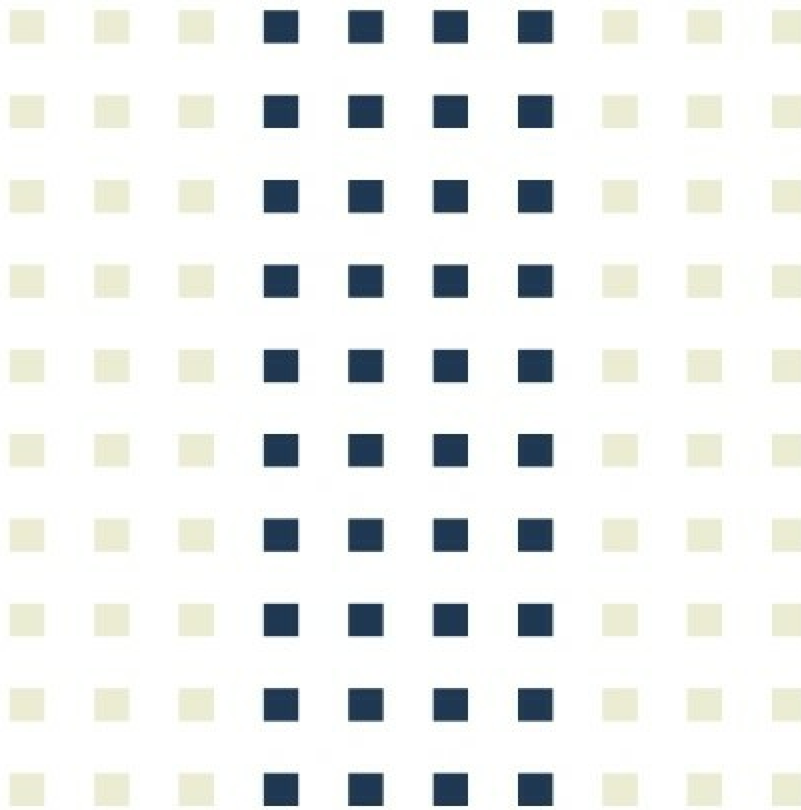


图9-20 数字1的点阵组成

根据规则，有数字的部分用1表示，空白的部分用−1表示。可以得到数字1的点阵表示如下。

```
one=
[-1,-1,-1,1,1,1,1,-1,-1,-1;...-1,-1,-1,1,1,1,1,-1,-1,-1;...-1,1,1,-1,-1,-1,-1,1,1,-1;...
```

注意：因为每个人的习惯不同，所以建立的数字点阵也可能不一样。

使用Hopfield神经网络识别数字1的MATLAB代码如下所示。

```
clear all
clc
%数据导入
```

```
load data0_9
%构建目标向量
T=[one]';
%创建Hopfield神经网络
net=newhop(T);
%随机产生带噪声的数字1点阵
noisy_one=one;
for i=1:100
    a=rand;
    if a<0.3
        noisy_one(i)=-one(i);
    end
end
%进行数字识别
noisy_one1={(noisy_one)'};
identify_one=sim(net,{10,10},{},noisy_one1);
identify_one{10}';
%结果显示
Array_one=imresize(one,20);
subplot(1,3,1)
imshow(Array_one)
title('标准(数字1)')
subplot(1,3,2)
Noisy_one=imresize(noisy_one,20);
imshow(Noisy_one)
title('噪声(数字1)')
subplot(1,3,3)
```

```
imshow(imresize(identify_one{10}',20))
title('识别(数字1)')
```

运行上述代码得到图9-21所示的结果。



图9-21 Hopfield神经网络识别数字1的结果

从图9-21中可以看出，尽管加入了噪声，Hopfield神经网络还是能准确地识别数字1。

9.4.3 DHNN的海布（Hebb）学习规则

在DHNN的网络训练过程中，运用的是海布调节规则：当神经元输入与输出节点的状态相同时，从第j个到第i个神经元之间的连接强度则增强，否则减弱。海布法则是一种无指导的死记式学习算法。

离散型Hopfield网络的学习目的，是对具有q个不同的输入样本组$P_{r\times q}$＝[P_1， P_2…P_q]，通过调节计算有限的权值矩阵W，系统能够收敛到各自输入样本矢量本身。当k＝1时，对于第i个神经元，由海布学习规则可得网络权值对输入矢量的学习关系式为

$$w_{ij} = \alpha p_j^1 p_i^1$$

其中，α＞0，i＝1，2…，r；j=1，2…，r。在实际学习规则的运用中，一般取α＝1或1/r。上式表明了海布调节规则：神经元输入P与输出A的状态相同（即同时为正或为负）时，从第j个到第i个神经元之间的

连接强度w_{ij}则增强（即为正）；神经元输入P与输出A的状态不相同（即一个为正，另一个为负）时，w_{ij}则减弱（即为负）。

对于由上式求出的权值w_{ij}是否能够保证$a_i = p_j$，可以取α＝l，我们来验证一下，对于第i个输出节点，有

$$a_i^1 = \text{sgn}(\sum_{j=1}^{r} w_{ij} p_j^1) = \text{sgn}(\sum_{j=1}^{r} p_j^1 p_i^1 p_j^1) = \text{sgn}(p_i^1) = p_i^1$$

因为p_i和a_i值均取二值{-l,1}，所以当其为正值时，即为1；其值为负值时，即为−1。同符号值相乘时，输出必为1。

由$\text{sgn}(p_i^1)$可以看出，不一定需要$\text{sgn}(p_i^1)$的值，只要符号函数$\text{sgn}(\bullet)$中的变量符号与p_i^1的符号相同，即能保证$\text{sgn}(\bullet) = p_i^1$。这个符号相同的范围就是一个稳定域。

当k＝1时，海布规则能够保证$a_i^1 = p_i^1$成立，使网络收敛到自己。现在的问题是，对于同一权矢量W，网络不仅要能够使一组输入状态收敛到其稳态值，而且是要能够同时记忆住多个稳态值，即同一个网络权矢量必须能够记忆住多组输入样本，使其同时收敛到不同对应的稳态值。

根据海布规则的权值设计方法，当k由1增加到2，直至q时，则是在原有己设计出的权值的基础上，增加一个新量$p_j^k p_i^k$，k＝2…,q，所以对网络所有输入样本记忆权值的设计公式为

$$w_{ij} = \alpha \sum_{k=1}^{q} t_j^k t_i^k$$

DHNN 的设计目的是使任意输入矢量经过网络循环最终收敛到网络所记忆的某个样本上。

因为Hopfield网络有$w_{ij}=w_{ji}$，所以完整的Hopfield网络权值设计公式应当为

$$w_{ij} = \alpha \sum_{\substack{k=1 \\ i \neq j}}^{q} t_j^k t_i^k$$

采用海布学习规则来设计记忆权值，是因为设计简单，并可以满足$w_{ij}=w_{ji}$的对称条件。从而可以保证网络在异步工作时收敛。

在同步工作时，网络或收敛或出现极限环为　2。在设计网络权值时，与前向网络不同的是令初始权值$w_{ij}=0$，每当一个样本出现时，都在原权值上加上一个修正量，即$w_{ij}=w_{ij}+t_j^k t_i^k$。

对于第k个样本，当第i个神经元输出与第j个神经元输入同时兴奋或同时抑制时，$t_j^k t_i^k$＞o；当$t_j^k t_i^k$中一个兴奋一个抑制时，$t_j^k t_i^k$＜0。这就和海布提出的生物神经细胞之间的作用规律相同。

在神经网络工具箱中有关采用海布公式求解网络权矩阵变化的函数为 learnh.m 和learnhd.m，后者为带有衰减学习速率的函数。

dW＝learnh(P,A,lr)；

或 dW＝learnhd(W,P,A,lr,dr)；

对于简单的情况，lr可以选择1；对于复杂的应用，可取lr＝0.1～

0.5，dr＝lr/3。

9.4.4 DHNN权值设计的其他方法

用海布规则设计出的DHNN权值能够保证其网络在异步工作时稳定收敛，尤其在记忆样本是正交的条件下，可以保证每个记忆样本收敛到自己，并有一定范围的吸引域，但对于那些不是正交的记忆样本，用此规则设计出的网络则不一定能收敛到本身。

这里介绍几种其他的权值设计方法以对此不足加以改进，它们各有自己的特点。

1.δ学习规则

δ学习规则基本公式为

$$\Delta W = \eta \cdot \delta \cdot P$$

$$w_{ij}(t+1) = w_{ij}(t) + \eta[T(t) - A(t)]P(t)$$

即通过计算每个神经元节点的实际激活值A(t)，与期望状态T(t)进行比较，若不满足要求，则将二者的误差的一部分作为调整量，若满足要求，则相应的权值保持不变。

2.伪逆法

对于输入样本P＝[P^1，P^2，…，P^q]，设网络输出可以写成一个与输入样本相对应的矩阵A，输入和输出之间可用一个权矩阵W 来映射，即有W·P＝N，A＝sgn(N)，由此可得

$$W=N\cdot P'$$

其中P'为P的伪逆，有$P' = (P^{\mathrm{T}}P)^{-1}P^{\mathrm{T}}$，如果样本之间是线性无关的，则$P^{\mathrm{T}}P$满秩，其逆存在，则可求出W =N·P'中的权矩阵W。

用伪逆法求出的权矩阵W，可以保证对所记忆的模式，在输入时仍能够正确收敛到样本自己，在选择A值时，只要满足A矩阵中的每一个

元素与W·P矩阵中的每个元素有相同的符号，甚至可以简单地选择A与P具有相同符号的值，即可满足收敛到学习样本的本身。

但当记忆样本之间是线性相关的，由海布法所设计出网络存在的问题，伪逆法也解决不了，甚至无法求解，相比之下，由于存在求逆等运算，伪逆法较为繁琐，而海布法则要容易求得多。

3.正交化的权值设计

这一方法的基本思想和出发点是为了满足下面4个要求。

·保证系统在异步工作时的稳定性，即它的权值是对称的，满足 $w_{ij}=w_{ji}$，i,j＝1,2,…,s。

·保证所有要求记忆的稳定平衡点都能收敛到自己。

·使伪稳定点的数目尽可能少。

·使稳定点的吸引域尽可能大。

正交化权值计算公式推导如下。

（1）已知有q个需要存储的稳定平衡点T^1, T^2, …,T^q, $T\in R^s$，计算s×(q−l)阶矩阵$YR^{s\times(q-1)}$：

$$Y=[T^1-T^q\ T^2-T^q\ \cdots T^{q-1}-T^q]^{\mathrm{T}}$$

（2）对Y进行奇异值及酉矩阵分解，如存在两个正交矩阵U和V以及一个对角值为Y的奇异值的对角矩阵A，满足

$$Y=UAV$$

$$Y=[T^1\ T^2\ \cdots T^{q-1}]^{\mathrm{T}}$$

$$U=[U^1\ U^2\ \cdots U^S]^{\mathrm{T}}$$

$$V=[V^1\ V^2\ \cdots V^{q-1}]^{\mathrm{T}}$$

$$A=\begin{bmatrix} \lambda_1 & & \cdots & & 0 \\ & \ddots & & \iddots & \\ \vdots & & \lambda_k & & \\ & \iddots & & \ddots & \\ 0 & & \cdots & & 0 \end{bmatrix}$$

k维空间为s维空间的子空间，它由k个独立基组织

k=rank(A)

设$\{U^1U^2\ldots U^k\}$为Y 的正交基，而$\{U^{k+1}U^{k+2}\ldots U^s\}$为s 维空间中的补充正交基，下面利用U矩阵来设计权值。

（3）定义。

$$W^+=\sum_{j=1}^{k}U^i(U^i)^{\mathrm{T}}$$

$$W^-=\sum_{j=k+1}^{k}U^i(U^i)^{\mathrm{T}}$$

总的连接权值为

$$W_\tau=W^+-\tau W^-$$

其中τ为大于−1的参数。

由此可见，网络的权矩阵是由两部分的权矩阵 W^+和 W^-相加而成的。这两部分权矩阵均满足对称条件，即有下式成立

$$w_{ij}^+=w_{ji}^+\quad ;\quad w_{ij}^-=w_{ji}^-$$

因而W_t中分量也满足对称条件。这就保证了系统在异步时能够收敛并且不会出现极限环。

9.5 连续型Hopfield网络

Hopfield网络可以推广到输入和输出都取连续数值的情形。这时网络的基本结构不变，状态输出方程形式上也相同。若定义网络中第i个神经元的输入总和为n_i，输出状态为a_i，则网络的状态转移方程可写为

$$a_i = f(\sum_{j=1}^{r} w_{ij} p_j + b_i)$$

其中神经元的激活函数f为S型的函数：

$$f = \tanh(\lambda(n_i + b_i))$$

通过下面MATLAB程序得到连续Hopfield神经网络的激活函数tanh图形如图9-22所示。

```
x =-5:0.01:5;
y = tanh(x);
plot(x,y)
grid on
```

连续型Hopfield 神经网络很适合求解TSP（Traveling Salesman Problem）问题。所谓TSP 问题，即“旅行商问题”是一个十分有名的难以求解的优化问题，其要求很简单：在n个城市的集合中，找出一条经过每个城市各一次，最终回到起点的最短路径。

【例9-5】应用连续型Hopfield神经网络求解TSP问题。

解：如果已知城市A，B，C，D，…，之间的距离为d_{AB}，d_{BC}，d_{CD}…；那么总的距离d＝d_{AB}+d_{BC}+d_{CD}+…，对于这种动态规化问题，要去求其min(d)的解。因为对于n个城市的全排列共有n种，而TSP并没

有限定路径的方向，即为全组合，所以对于固定的城市数n的条件下，其路径总数S_n为S_n＝n!/2n (n≥4)，例如n＝4 时，S_n＝3，即有3 种方式，如图9-23所示。

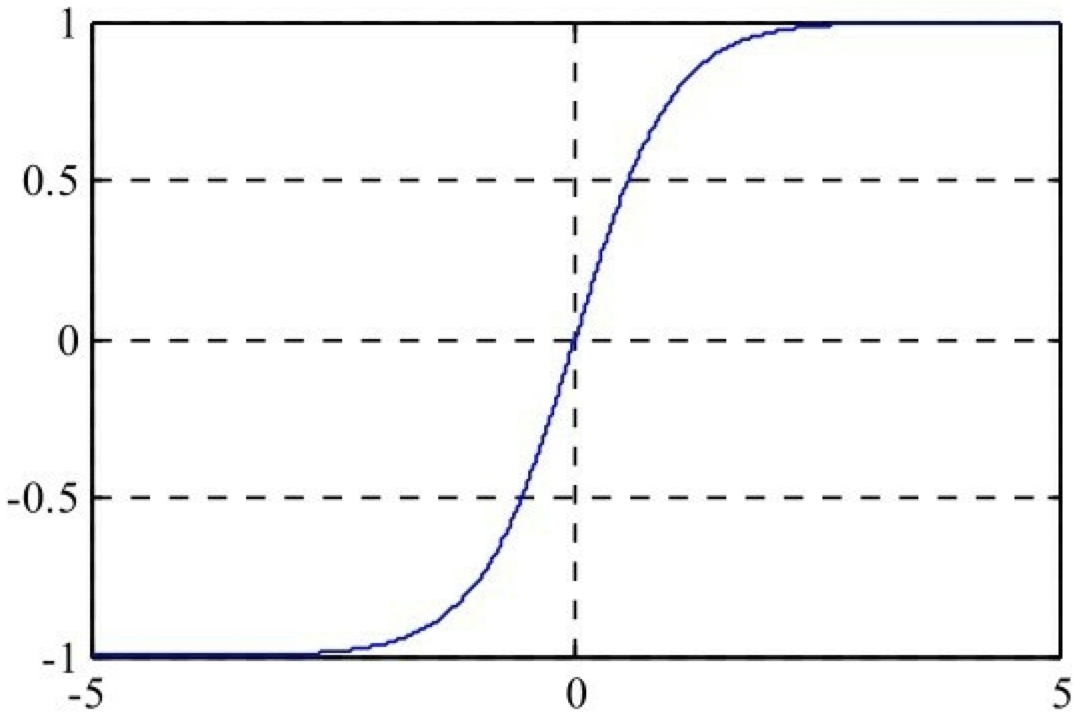


图9-22 连续Hopfield网络激活函数

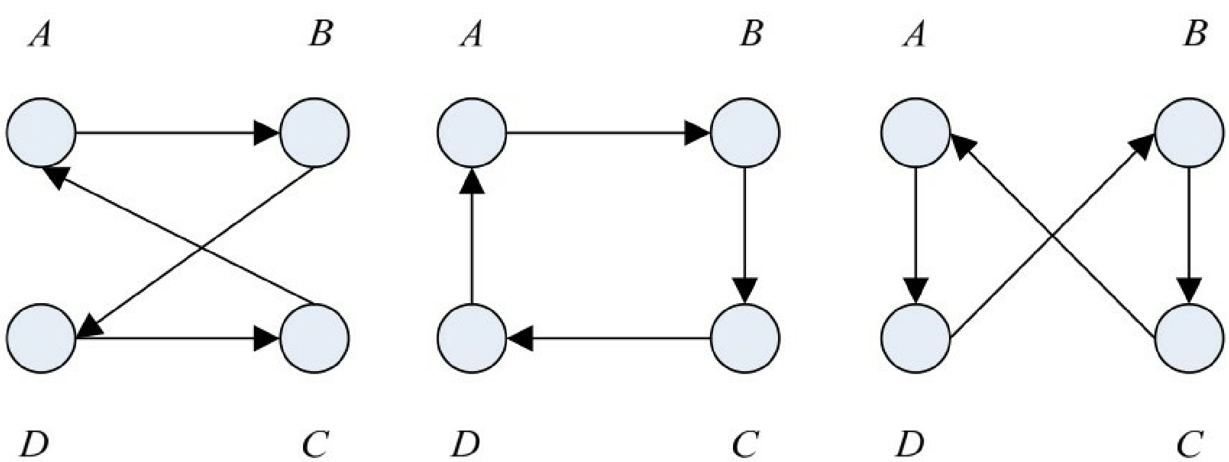


图9-23 n＝4时的TSP路径图

表9-1 城市数和对应的旅行方案数

城 市 数	旅行方案数=$n!/2n$
3	1
4	3
5	12
6	60
7	360
8	2 520
9	20 160
10	181 440
11	19 958 400
12	239 500 800
…	…

由斯特林（Stirlin）公式，路径总数可写为

$$S_n = \frac{1}{2n}[\sqrt{2\pi n} \cdot \mathrm{e}^{n(\ln(n-1))}]$$

若采用穷举搜索法，则需要考虑所有可能的情况，找出所有的路径，再分别对其进行比较，以找出最佳路径，因其计算复杂程度随城市数目的增加呈指数增长，可能达到无法进行的地步。

从表 9-1 中可以看到，当城市数为 12 时，旅行方案数已达 239 500 800 种。而每增加一个城市，所增加的方案数为

$$\frac{(n+1)!}{2(n+1)} \Big/ \frac{n!}{2n} \approx n$$

这类问题称为完全非确定性多项式问题（Nondeterministic Polynomial Complete，简称NP完全问题）。

由于求解最优解的负担太重，通常比较现实的做法是求其次优解。Hopfield 网络正是一种合适的方法。因为它可以保证其解向能量函数的最小值方向收敛，但不能确保达到全局最小值点。

Hopfield神经网络的设计步骤如图9-24所示。



图9-24 Hopfield神经网络设计步骤

采用连续时间的Hopfield网络模型来求解TSP，开辟了一条解决这一问题的新途径。其基本思想是把TSP映射到CHNN上，通过网络状态的动态演化逐步趋向稳态而自动地搜索出优化解。

1.模型映射

为了便于神经模型的实现，必须首先找到过程的一个合适的表达方法。TSP 的解是若干城市的有序排列，任何一个城市在最终路径上的位置可用一个n维的0、1矢量表示，对于所有n个城市，则需要一个n×n维矩阵。

以5个城市为例，一种可能的排列矩阵见表9-2。

表9-2 5个城市可能的排列矩阵

	1	2	3	4	5
A	0	1	0	0	0
B	0	0	0	1	0
C	1	0	0	0	0
D	0	0	1	0	0
E	0	0	0	0	1

其中，行矢量表示城市名，列矢量表示城市中在旅行中排的序号。该矩阵唯一地确定了一条有效的行程路径

$$C\rightarrow A\rightarrow D\rightarrow B\rightarrow E$$

很明显，为了满足约束条件，该矩阵中每一行以及每一列中只能有一个元素为 1，其余元素均为零。这个矩阵称为关联阵。若用d_{xy}表示从城市x到城市y的距离，则上面路径的总长度为

$$d_{xy}=d_{CE}+d_{AD}+d_{DB}+d_{BE}$$

TSP的最优解是求长度d_{xy}为最短的一条有效的路径。为了解决TSP，必须构造这样一个网络：在网络运行时，其能量能不断降低；在运行稳定后，网络输出能代表城市被访问的次序，即构成上述的关联矩

阵。

2.构造网络能量函数

网络能量的最小值对应于最佳（或次最佳）的路径距离。所以解决问题的关键，仍然是构造合适的能量函数。

对于一个n城市的TSP，需要n×n节点的CHNN。假设每个神经元的输出记为V_{xi}，　V_{yi}，行下标x和y表示不同的城市名，列下标i和j表示城市在路径中所处的次序位置，通过V_{xi}，V_{yi}取0或1，可以通过关联矩阵确定出不同种的访问路径。用d_{xy}表示两个不同的城市之间的距离，对于选定的任一V_{xi}，和它相邻的另一个城市y的状态可以有$V_{y(i+1)}$和$V_{y(i-1)}$。那么，目标函数f(V)可选为

$$f(V)=\frac{D}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{i=1}^{N}d_{xy}V_{xi}V_{y(i+1)}$$

这里所选择的 f(V)表示的是对应于所经过的所有路径长度的总量，其数值为一次有效路径总长度的倍数，当路径为最佳时，f(V)达到最小值，它是输出的函数。

当V_{xi}＝0时，则有f(V)＝0，此输出对f(V)没有贡献；当V_{xi}＝1时，则通过与i相邻位置的城市i+1和i−1的距离，如在关联矩阵中V_{D3}＝1，那么，与i＝3相邻位上的两个城市分别为V_{A2}和V_{B4}，此时在f(V)中可得到d_{AD}和d_{DB}两个相加的量，依此类推，把旅行商走过的全部距离全加起来，即得f(V)。

约束条件要保证关联矩阵的每一行每一列中只有一个值为 1，其他值均为零，用两项表示为

$$g(V)=\frac{A}{2}\sum_{x=1}^{N}(\sum_{i=1}^{N}V_{xi}-1)^2+\frac{A}{2}\sum_{i=1}^{N}(\sum_{x=1}^{N}V_{xi}-1)^2$$

总的能量函数E为

$$
\begin{aligned}
E &= g(V) + f(V) \\
&= \frac{A}{2}\sum_{x=1}^{N}\left(\sum_{i=1}^{N}V_{xi}-1\right)^2 + \frac{A}{2}\sum_{i=1}^{N}\left(\sum_{x=1}^{N}V_{xi}-1\right)^2 + \frac{D}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{i=1}^{N}d_{xy}V_{xi}V_{y(i+1)}
\end{aligned}
$$

实现能量函数的函数代码如下所示。

```
%%%%%%计算能量函数%%%%%%%%%%%%
function E=energy(V,d)
global A D
n=size(V,1);
sum_x=sumsqr(sum(V,2)-1);
sum_i=sumsqr(sum(V,1)-1);
V_temp=V(:,2:n);
V_temp=[V_temp V(:,1)];
sum_d=d*V_temp;
sum_d=sum(sum(V.*sum_d));
E=0.5*(A*sum_x+A*sum_i+D*sum_d);
```

Hopfield神经网络动态方程为

$$
\frac{dU_{xi}}{dt} = -\frac{\partial E}{\partial V_{xi}} = -A\left(\sum_{i=1}^{N}V_{xi}-1\right) - A\left(\sum_{y=1}^{N}V_{yi}-1\right) - D\left(\sum_{y=1}^{N}d_{xy}V_{y(i+1)}\right)
$$

实现动态方程的函数代码如下所示。

```
%%%%%%%%%计算du%%%%%%%%%%%
function du=diu(V,d)
global A D
n=size(V,1);
sum_x=repmat(sum(V,2)-1,1,n);
sum_i=repmat(sum(V,1)-1,n,1);
```

```
V_temp=V(:,2:n);
V_temp=[V_temp V(:,1)];
sum_d=d*V_temp;
du=-A*sum_x-A*sum_i-D*sum_d;
```

3.初始化网络

用连续Hopfield网络求解像TSP这样约束优化问题时，系统参数的取值对求解过程有很大影响。霍普菲尔德和泰克（Tank）经过实验，认为取初始值为 A＝500，D＝200 时，其求解10个城市的TSP得到良好的效果。

网络输入初始化选取

$$U_{xi}(t)=U_0\ln(N-1)+\delta_{xi}\quad (x,i=1,2,\cdots,N;t=0)$$

其中，U_0＝0.2，N为城市个数10，δ_{xi}为（-1,1）区间的随机值。

在本次网络迭代过程中，采样时间设置为0.0005，迭代次数为5000。

4.优化计算

当Hopfield神经网络的结构和参数确定后，迭代优化计算的过程就变得非常简单。

解决本题TSP问题的MATLAB代码如下。

```
clear all
clc
%定义全局变量
global A D
%导入城市位置
load location
%计算相互城市间距离
distance=dist(citys,citys');
```

```
%初始化网络
N=size(citys,1);
A=500;
D=200;
U0=0.2;
step=0.00005;
delta=2*rand(N,N)-1;
U=U0*log(N-1)+delta;
V=(1+tansig(U/U0))/2;
iter_num=5000;
E=zeros(1,iter_num);
%寻优迭代
for k=1:iter_num
    % 动态方程计算
    dU=diu(V,distance);
    % 输入神经元状态更新
    U=U+dU*step;
    % 输出神经元状态更新
    V=(1+tansig(U/U0))/2;
    % 能量函数计算
    e=energy(V,distance);
    E(k)=e;
end
%判断路径有效性
[rows,cols]=size(V);
V1=zeros(rows,cols);
[V_max,V_ind]=max(V);
```

```
    for j=1:cols
        V1(V_ind(j),j)=1;
    end
    C=sum(V1,1);
    R=sum(V1,2);
    flag=isequal(C,ones(1,N)) & isequal(R',ones(1,N));
    %结果显示
    if flag==1
        % 计算初始路径长度
        sort_rand=randperm(N);
        citys_rand=citys(sort_rand,:);
        Length_init=dist(citys_rand(1,:),citys_rand(end,:)');
        for i=2:size(citys_rand,1)
            Length_init=Length_init+dist(citys_rand(i-1,:),citys_rand(i,:)');
        end
        % 绘制初始路径
        figure(1)
        plot([citys_rand(:,1);citys_rand(1,1)],
[citys_rand(:,2);citys_rand(1,2)],'o-')
    for i=1:length(citys)
            text(citys(i,1),citys(i,2),[' ' num2str(i)])
        end
        text(citys_rand(1,1),citys_rand(1,2),['    起点' ])
        text(citys_rand(end,1),citys_rand(end,2),['    终点' ])
        title(['优化前路径(长度：' num2str(Length_init) ')'])
        axis([0 1 0 1])
        grid on
```

```
xlabel('城市位置横坐标')
ylabel('城市位置纵坐标')
% 计算最优路径长度
[V1_max,V1_ind]=max(V1);
citys_end=citys(V1_ind,:);
Length_end=dist(citys_end(1,:),citys_end(end,:)');
for i=2:size(citys_end,1)
    Length_end=Length_end+dist(citys_end(i-1,:),citys_end(i,:)');
end
% 绘制最优路径
figure(2)
plot([citys_end(:,1);citys_end(1,1)],...
    [citys_end(:,2);citys_end(1,2)],'o-')
for i=1:length(citys)
    text(citys(i,1),citys(i,2),['  ' num2str(i)])
end
text(citys_end(1,1),citys_end(1,2),['    起点' ])
text(citys_end(end,1),citys_end(end,2),['    终点' ])
title(['优化后路径(长度：' num2str(Length_end) ')'])
axis([0 1 0 1])
grid on
xlabel('城市位置横坐标')
ylabel('城市位置纵坐标')
% 绘制能量函数变化曲线
figure(3)
plot(1:iter_num,E);
ylim([0 1000])
```

```
    title(['能量函数变化曲线(最优能量：' num2str(E(end)) ')']);
    xlabel('迭代次数');
    ylabel('能量函数');
else
    disp('寻优路径无效');
end
```

5.仿真结果分析

运行MATLAB代码，随机产生的初始路径如图9-25所示。

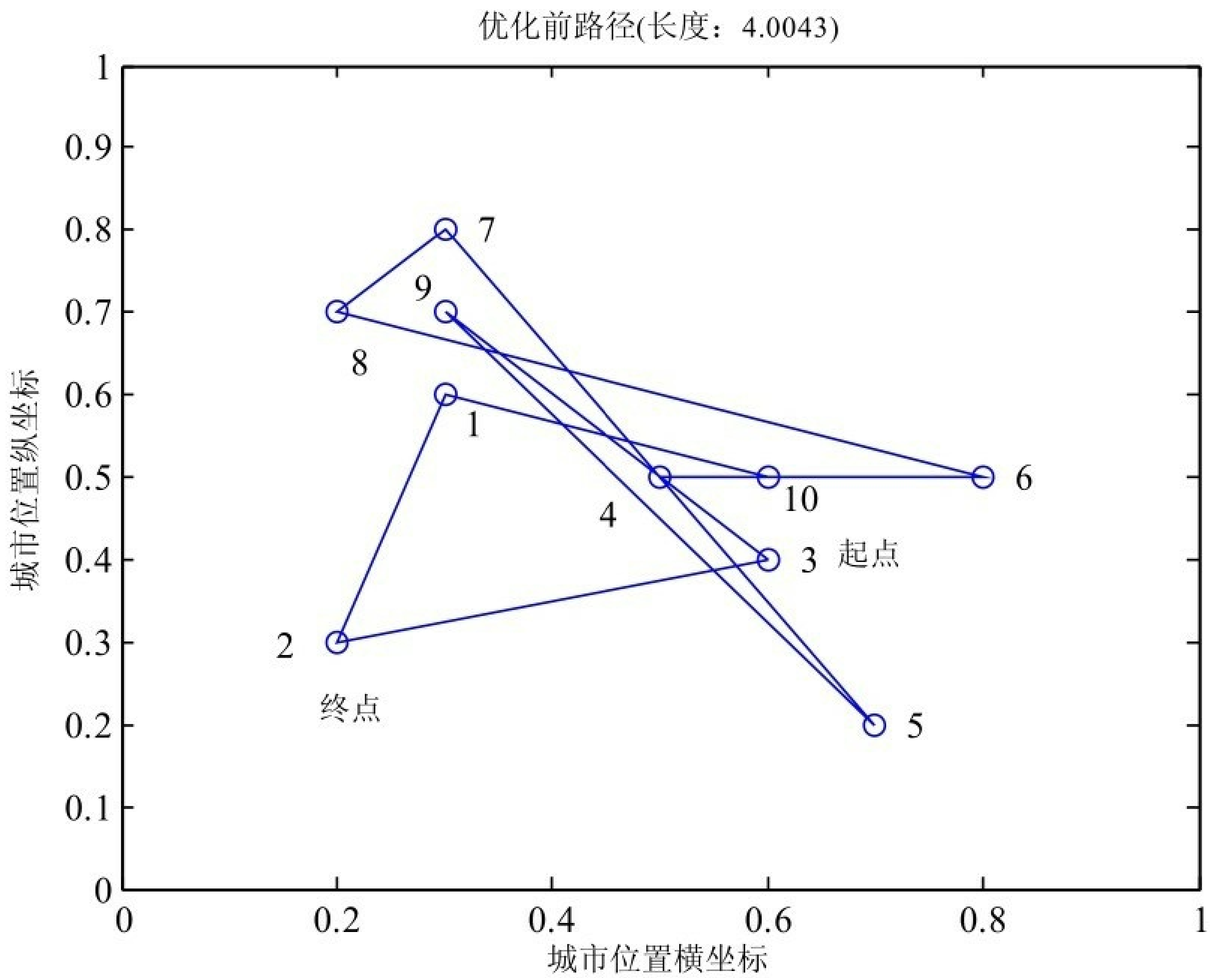


图9-25 随机产生的初始路径

由图9-25可知，随机产生的路径为 $3 \to 4 \to 9 \to 5 \to 7 \to 8 \to 6 \to 10 \to 1 \to 2$，其长度为4.0043。

经过连续型Hopfield神经网络优化后，查找到的优化路径如图9-26所示，优化路径具体为4→5→6→3→10→9→7→8→1→2，其长度为2.4206。

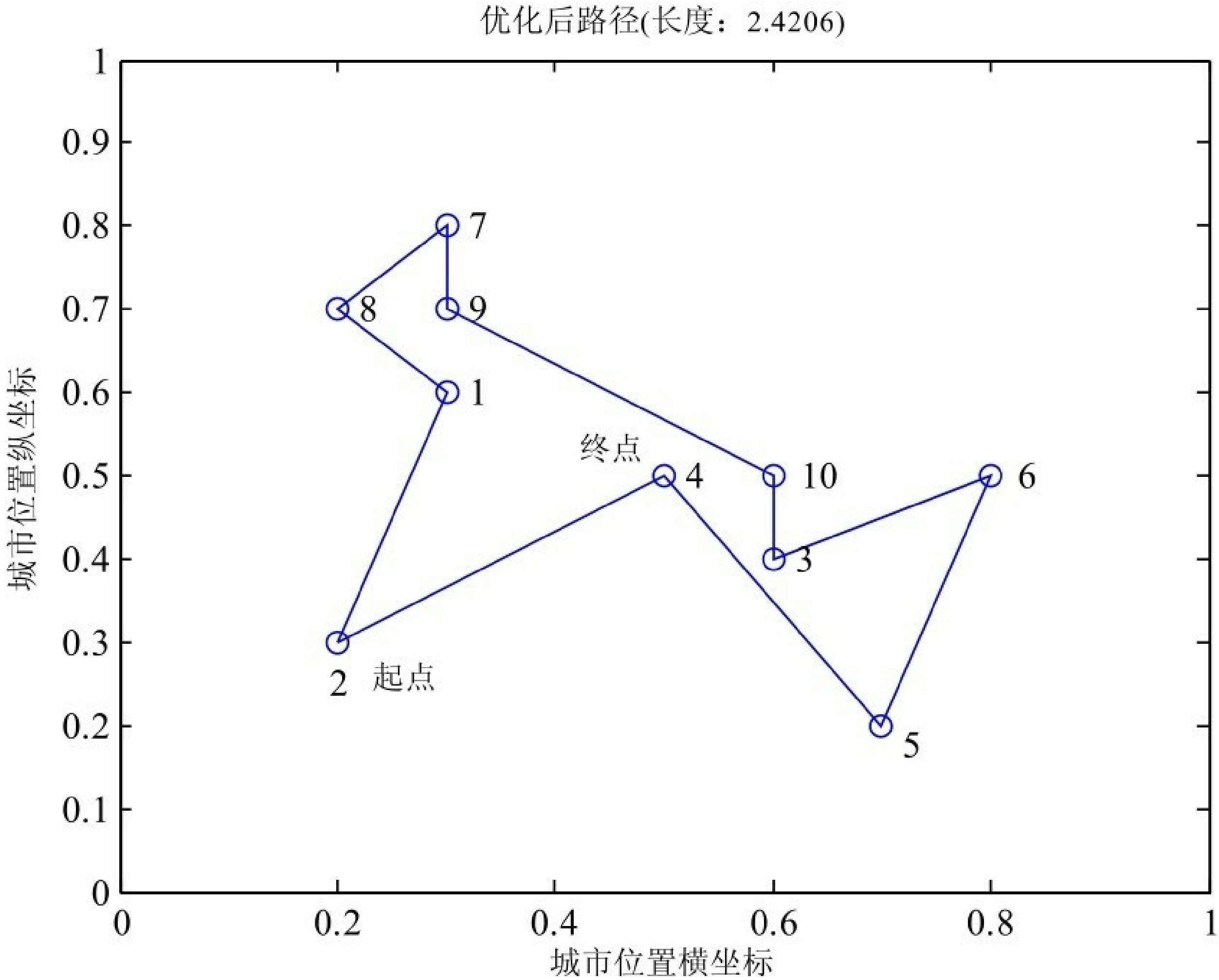


图9-26 优化路径

能量函数随迭代过程变化的曲线如图9-27所示。

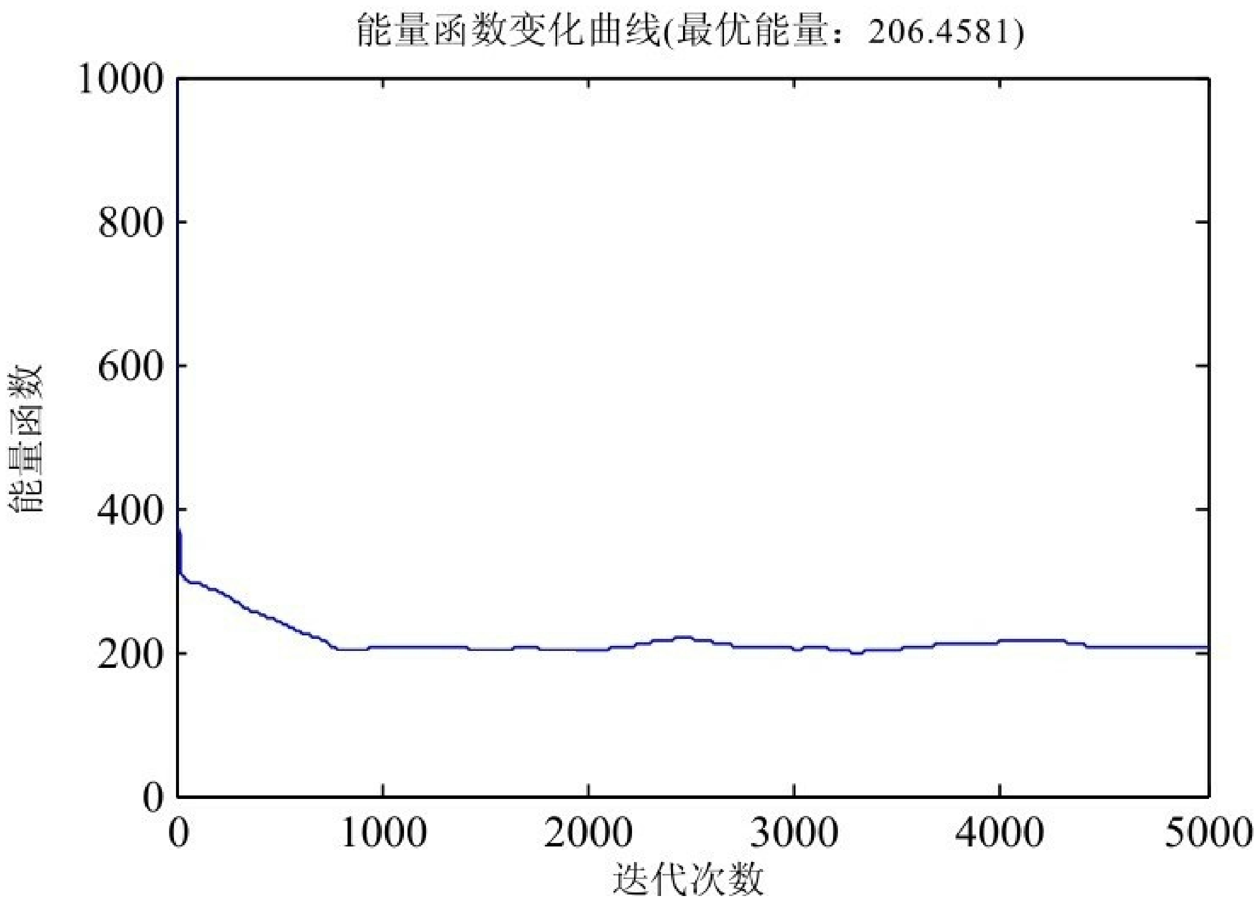


图9-27 能量函数随迭代过程变化的曲线

从图中可以看出，网络的能量随着迭代不断减少。当网络能量小到一定程度后，网络神经元的状态也趋于平衡点，此时对应的城市顺序即为待求的优化路径。

提示：因为函数中有随机数，所以每次运行程序得到的结果不一定完全一致。

9.6 Elman网络

Elman神经网络是Elman于1990年首先针对语音处理问题而提出来的。该模型在BP网络的隐含层中增加一个承接层，作为一步延时算子，达到记忆的目的，从而使系统具有适应时变特性的能力，能直接反映动态过程系统的特性。

9.6.1 Elman网络结构

Elman神经网络是应用较为广泛的一种典型的反馈型神经网络模型。一般分为4层：输入层、隐含层、连接层和输出层，如图9-28所示。

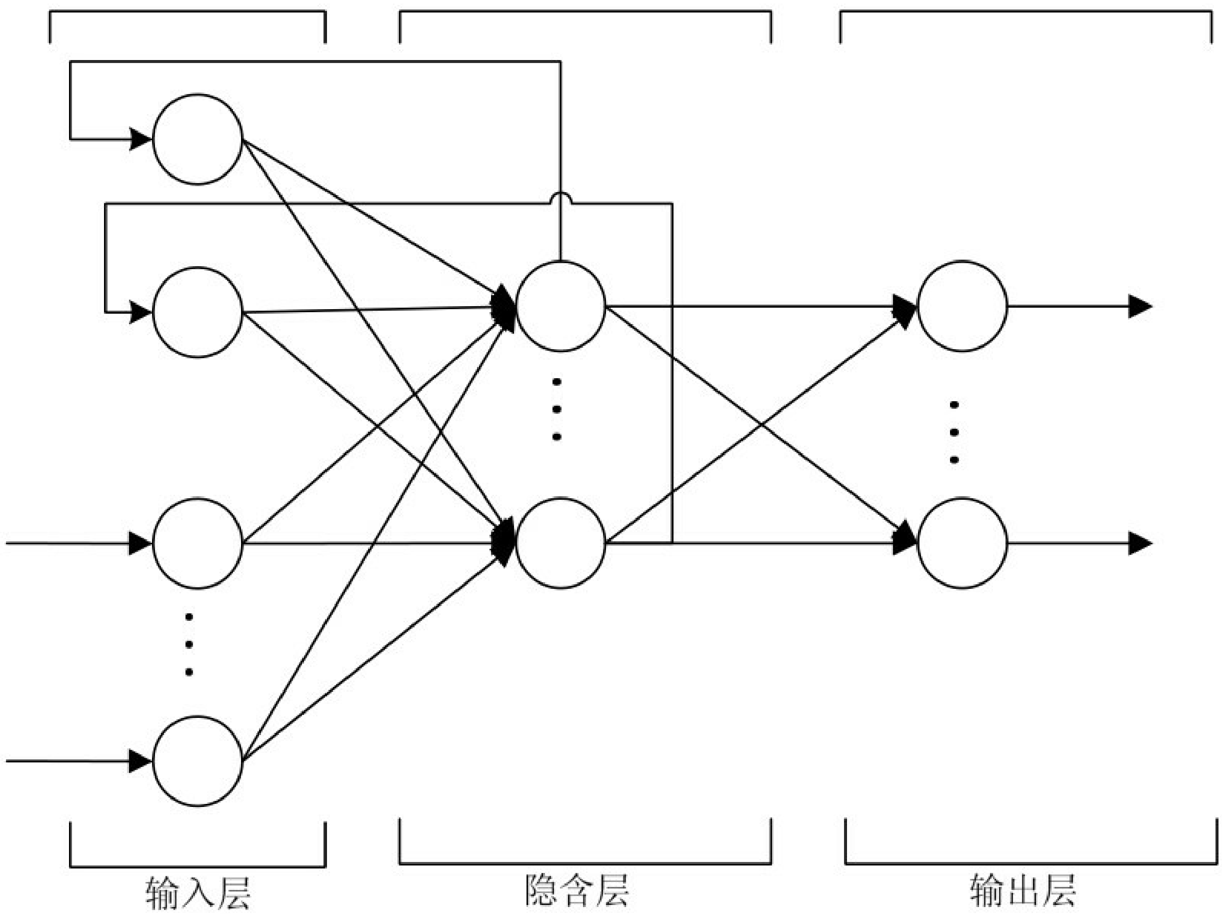


图9-28 Elman网络结构

Elman 型回归神经元网络的特点是隐含层的输出通过承接层的延迟与存储，自联到隐含层的输入，这种自联方式使得对历史状态的数据具有敏感性，内部反馈网络的加入又增加了网络本身处理动态信息的能力，从而达到了动态建模的目的。

9.6.2 Elman网络创建函数

网络创建函数为newelm，该函数的调用格式如下所示。

net = newelm([0 1],[5 1],{‘tansig’,‘logsig’})

以newelm定义的Elman网络为例，其反向传播训练算法的默认函数为trainbfg，还可以用trainlm，但其处理速度太快，这在Elman网络中是不必要的，效果也不一定好。默认的权值和阈值的反向传播函数为

leargdm，默认的误差性能函数为mse。

网络建立后，每个网络层的权值和阈值都以Nguyen-Widroe网络层初始化方法进行初始化，实现函数为initnw。

Elman网络的训练可采用train或adapt两个函数中的任意一个。当采用函数train时，每一步迭代过程按照以下步骤进行。

（1）在网络输入端先输入所有的输入序列，然后计算输出结果，并与目标序列进行比较，从而产生一误差序列。

（2）在每一次迭代中，误差被反向传播，以确定每一个权值和阈值的误差梯度，实际上梯度的计算是近似的，因为经由延迟反馈支路对权值和阈值误差的贡献是忽略的。

（3）该梯度用于对用户选择的反向传播训练函数进行权值修正。建议采用训练函数traingdx。

当采用函数adapt时，迭代过程与采用函数train时基本一样，只是权值的修正函数建议采用学习函数learngdm。

Elman网络没有其他网络的可靠性高，因为训练或调整都是以误差梯度的近似值进行的。

【例9-6】使用newelm建立一个Elman网络，并计算网络仿真输出与目标值之间的误差。

解：MATLAB代码程序如下。

```
clear all
clc
P = round(rand(1,20));
T = [0 (P(1:end-1)+P(2:end) == 2)];
Pseq = con2seq(P);
Tseq = con2seq(T);
net = newelm(P,T,10);
net = train(net,Pseq,Tseq);
```

```
Y = sim(net,Pseq);
z = seq2con(Y);
z{1,1};
diff1 = T - z{1,1};
plot(diff1)
```

在运行代码的时候，程序会显示神经网络的训练参数窗口，如图9-29所示。

由图9-29 所示可以看出，网络运行1000 步后，网络误差达到0.000105。

网络仿真输出与目标值之间的误差曲线如图9-30所示。

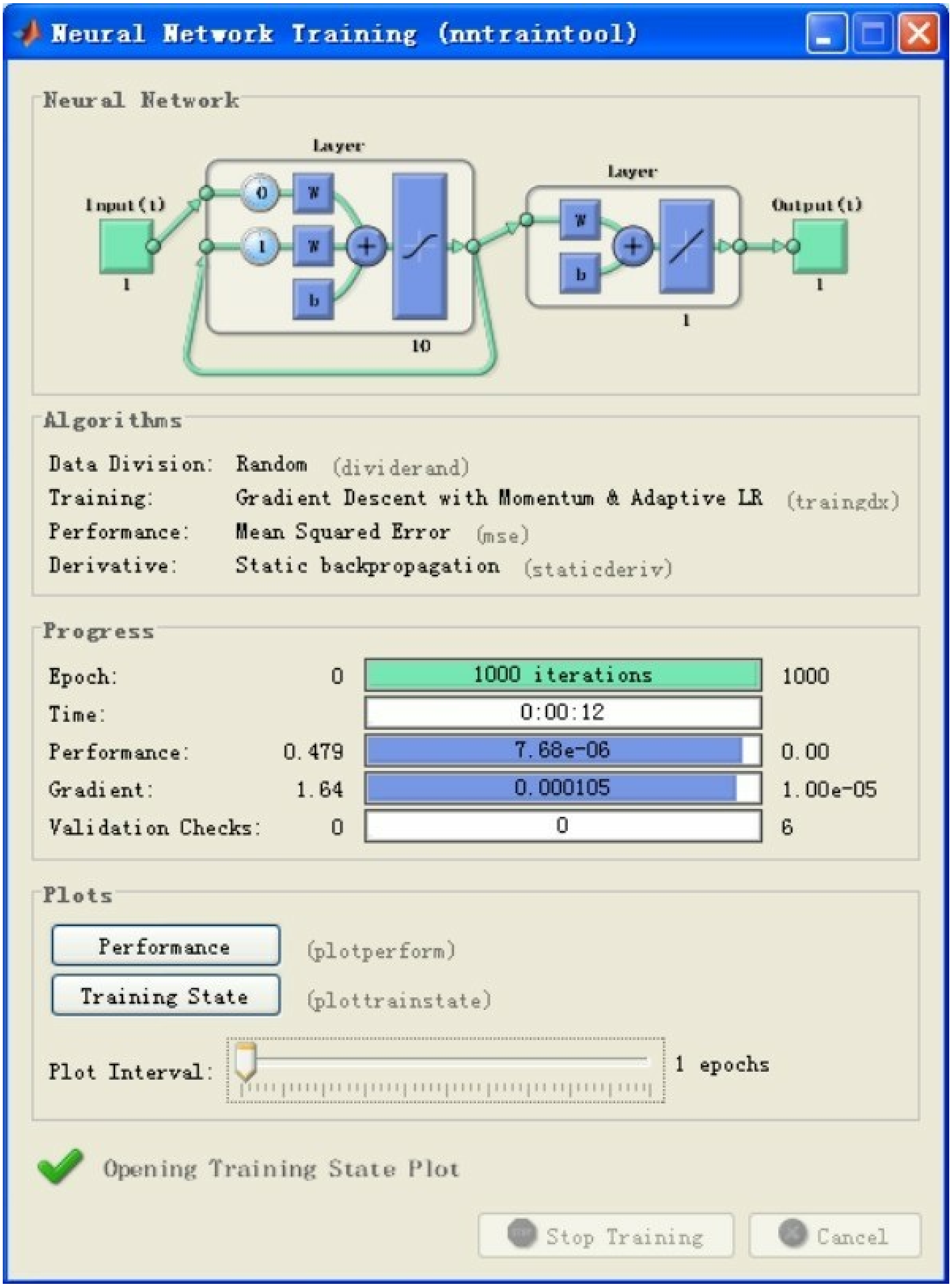


图9-29 神经网络的训练参数窗口

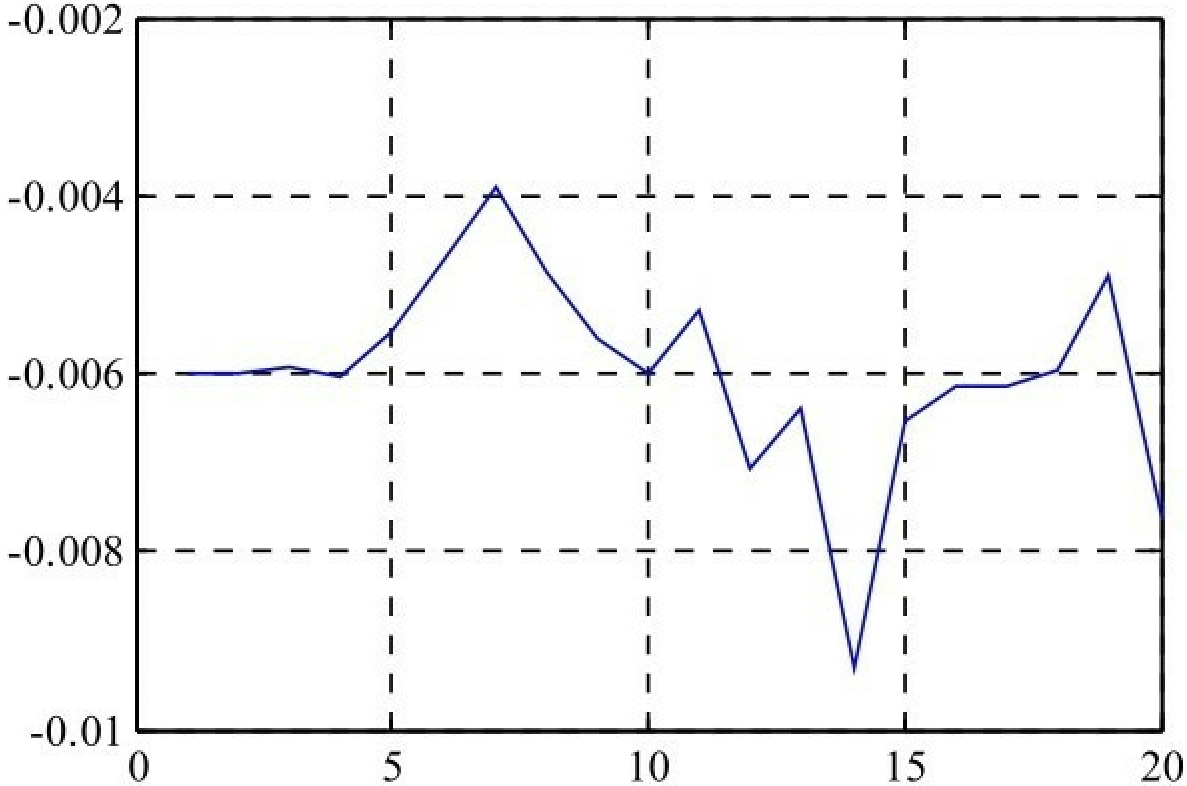


图9-30 网络仿真输出与目标值之间的误差曲线

由图9-30可知，网络的仿真输出与目标值之间最大误差为−0.004，表明建立的Elman网络预测能力较好。

9.6.3 Elman网络的工程应用

1.峰值检波

振幅调制是通信系统一种常见的模拟通信方式，在接收端往往采用峰值检波，通常采用峰值检波电路。

峰值检波电路一般是由一个运放构成的电压跟随器和二极管和电容构成当输入信号为正半周时二极管导通对电容充电，一直充电到峰值即最大值，当输入电压负半周时二极管截止，电容不放电，保持电压（峰值电压），这样电容两端电压一直处于峰值，可以检测出信号的峰值，称其为峰值检波。

Elman 神经网络是两层反向传播的网络，隐含层和输入向量连接的

神经元，其输出不仅作为输出层的输入，而且还连接隐含层内的另外一些神经元，反馈至隐含层的输入。

由于其输入表示了信号的空域信息，而反馈支路是一个延迟单元，反映了信号的时域信息，所以Elman网络可以在时域和空域进行模式识别。

【例9-7】将Elman神经网络用于峰值滤波中，对信号的峰值进行识别，建立Elman神经网络，并用MATLAB验证其准确性。

解：从Elman神经网络的机理上看，可以将AM已调波信号的输入看成是时域中的信号，网络在时域中先对其进行识别；而已调波的包络可以看成二维平面上的曲线，即为空域中的信号模式，Elman神经网络在空域中对输入向量的模式分类成为峰值检波的输出。

Elman神经网络的输入向量和目标向量曲线如图9-31所示。

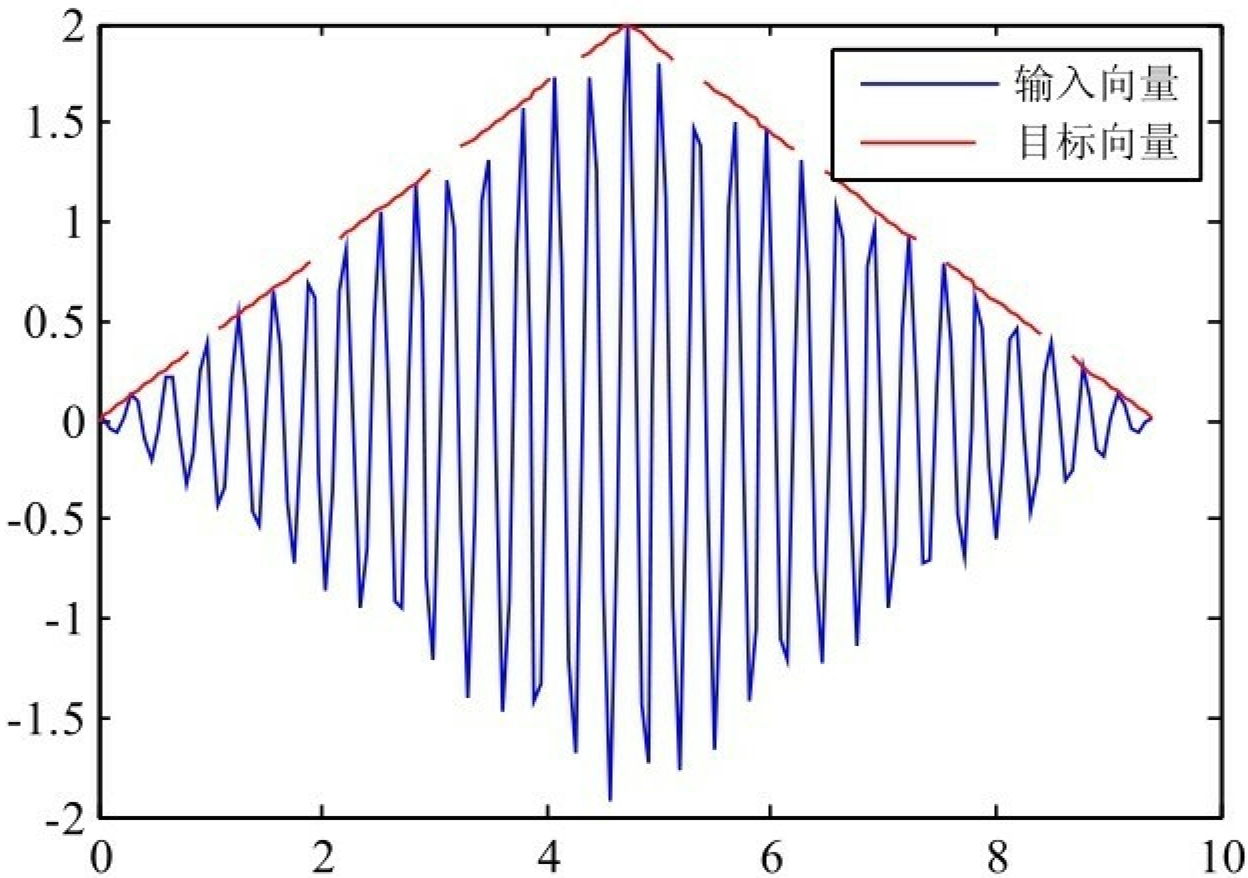


图9-31 Elman神经网络的输入向量和目标向量曲线

设计Elman神经网络的MATLAB代码如下。

```
clear all
clc
%%%%%%定义输入向量和目标向量%%%%%%%%%
time1=0:0.06:(1.5*pi);
t1=time1/(1.5*pi)-0.5;
time2=1.5*pi:0.06:(3*pi);
t2=1.5-time2/(1.5*pi);
time=[time1 time2];
t=2*[t1 t2];%目标向量
p=(1+t).*cos(20*time);%输入向量
%%%%%创建Elman网络%%%%%%%%%%%%%%%%%
%创建网络
net=newelm(minmax(p),[20 1],{'tansig','purelin'},'traingdx');
%训练网络
%将输入向量矩阵转换为输入序列
Pseq=con2seq(p);
%将目标向量矩阵转换为目标序列
Tseq=con2seq(t);
figure(1)
%画出调制信号和已调波形
plot(time,p,time,1+t,'r--')
hold on
pause;
net.trainParam.epochs=1000;
[net,tr]=train(net,Pseq,Tseq);
%%%%%%仿真%%%%%%%%%%%%%%%%%%%%%%
```

```
%以三角波进行仿真
time1=0:0.05:2*pi;
t1=time1/(2*pi)-0.5;
time2=2*pi:0.05:4*pi;
t2=1.5-time2/(2*pi);
time=[time1 time2];
%目标向量
t=[t1 t2];
%形成三角波调制的已调波信号
p=(1+t).*cos(20*time);
figure(2)
subplot(1,2,1);
%绘制三角波调制的已调波信号
plot(time,p)
%将输入向量矩阵转换为输入序列
Pseq=con2seq(p);
%网络仿真
a=sim(net,Pseq);
%将输出序列转换为矩阵形式
y=seq2con(a);
subplot(1,2,2);
%绘制网络输出信号波形和调制信号波形
plot(time,y{1},time,t,'r--');
hold on
%以正弦波调制进行仿真
t=0.5*sin(time);
%形成正弦波调制的已调波信号
```

```
p=(1+t).*cos(20*time);
figure(3)
subplot(1,2,1);
%绘制正弦波调制的已调信号
plot(time,p);
Pseq=con2seq(p);
a=sim(net,Pseq);
y=seq2con(a)
subplot(1,2,2);
%绘制网络输出信号波形和调制信号波形
plot(time,y{1},time,t,'r--');
hold on
%以矩形波调制进行仿真
t=0.5*sign(sin(time));
%形成矩形波调制的已调波信号
p=(1+t).*cos(20*time);
figure(4)
subplot(1,2,1);
%绘制矩形波调制的已调信号
plot(time,p);
Pseq=con2seq(p);
a=sim(net,Pseq);
y=seq2con(a)
subplot(1,2,2);
%绘制网络输出信号波形和调制信号波形
plot(time,y{1},time,t,'r--');
hold on
```

运行以上代码，得到训练网络的误差曲线如图9-32所示。从图中可以看出，经过1000步的训练，网络均方误差为0.002 473 1

以三角波进行仿真，结果如图9-33所示。

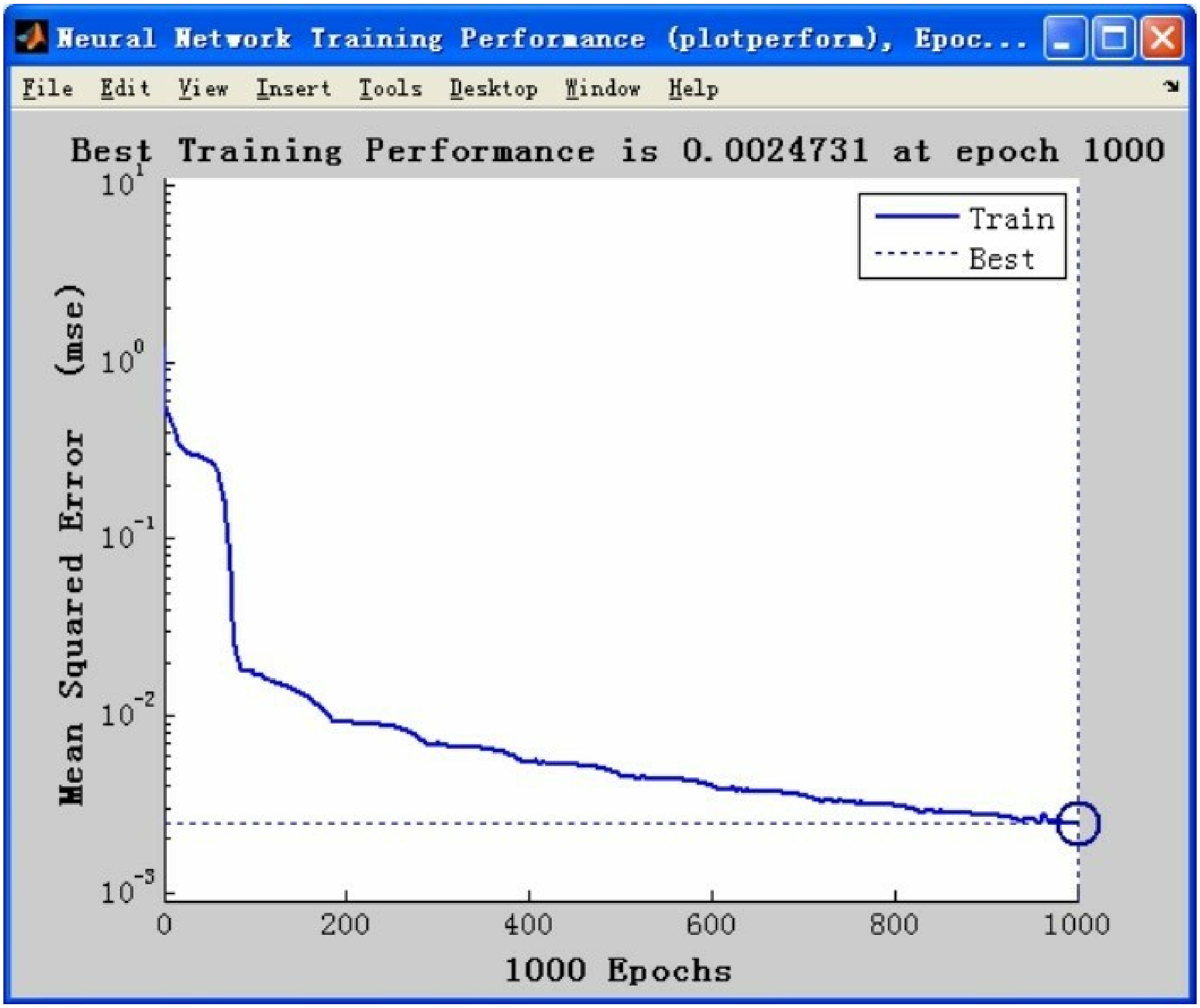


图9-32 训练网络的误差曲线

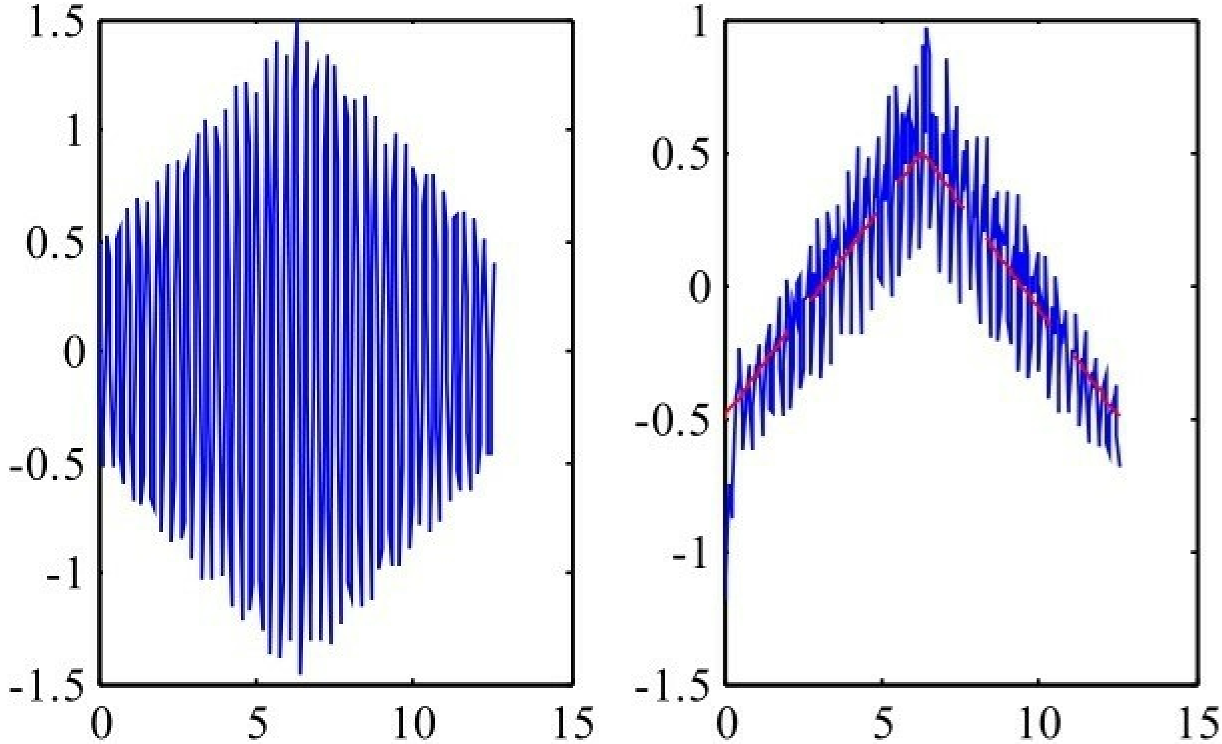


图9-33 三角波仿真结果

以正弦波进行仿真，结果如图9-34所示。

以矩形波进行仿真，结果如图9-35所示。

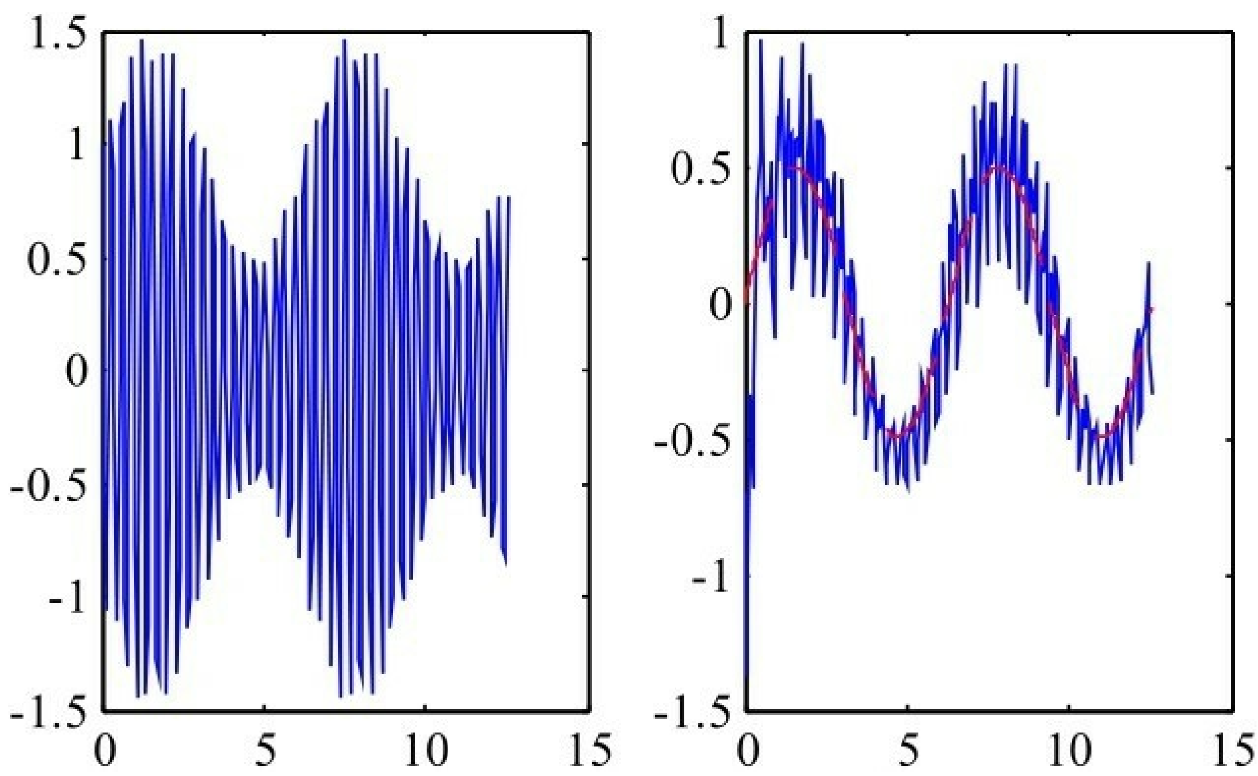


图9-34 正弦波仿真结果

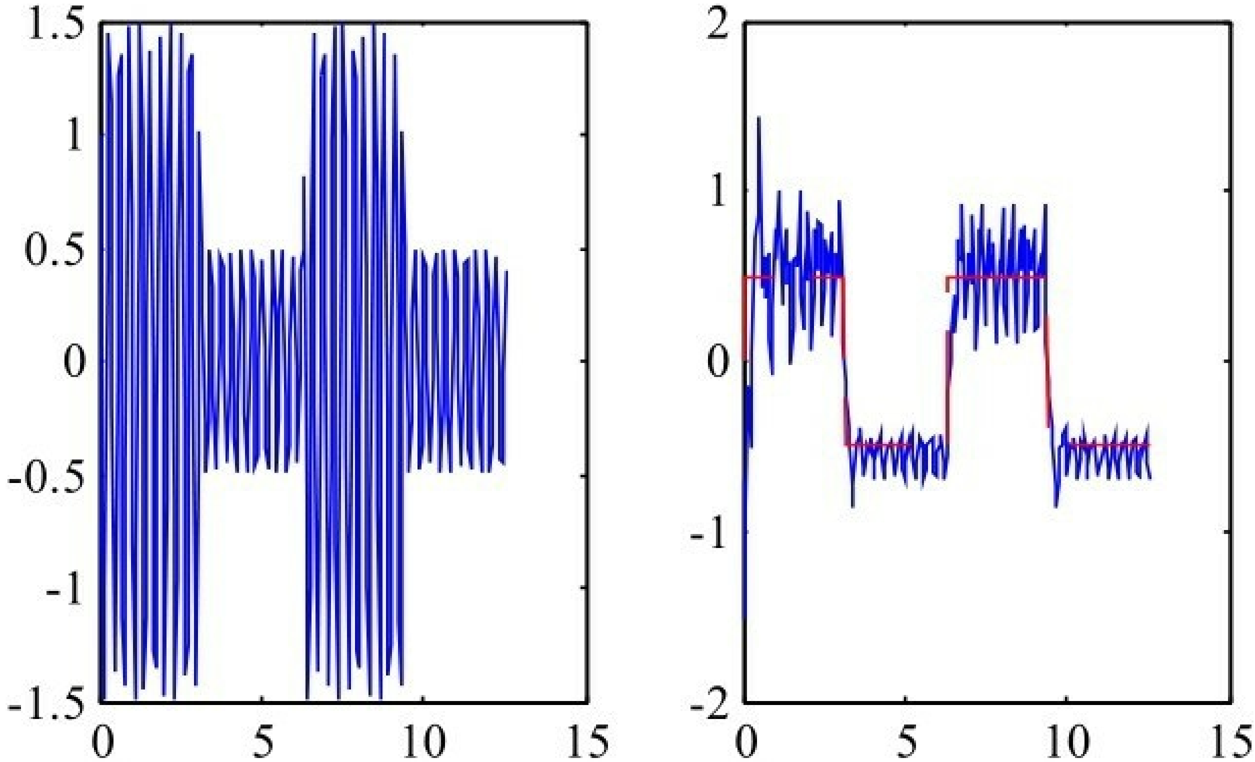


图9-35 矩形波仿真结果

在图9-33、图9-34和图9-35中，虚线是调制信号，实线是输出信号，调制信号和输出信号的变化趋势基本一致，即设计的Elman网络可以很好地完成不同调制信号、不同调幅度的峰值检波，输出波形中的纹波可以通过低通滤波器滤除。

2.电力负荷预测

电力负荷预测是指，在考虑一些重要的系统运行特性、增容决策和自然条件下，利用一套系统处理过去和未来负荷的方法，在一定精度意义下，决定未来某特定时刻或某些特定时刻的负荷值。

神经网络理论是利用神经网络的学习功能，让计算机学习包含在历史负荷数据中的映射关系，再利用这种映射关系预测未来负荷。由于该方法具有很强的鲁棒性、记忆能力、非线性映射能力以及强大的自学习能力，因此有很大的应用市场。

【例9-8】某10天内3个时间点的电网负荷见表9-3，利用Elman网络预测第11天对应时间内的电力负荷。

表9-3 10天内3个时间点的电网负荷

天　　数	负 荷 数		
1	0.37	0.51	0.71
2	0.12	0.17	0.88
3	0.32	0.99	0.69
4	0.13	0.55	0.63
5	0.11	0.42	0.84
6	0.24	0.53	0.71
7	0.33	0.44	0.9
8	0.19	0.66	0.44
9	0.31	0.67	0.49
10	0.37	0.51	0.71

解：在表9-3的数据中，选取前面9天的数据作为神经网络的训练样本，每3天的负荷作为输入向量，第4天的负荷作为目标向量。这样可以得到6组训练样本。第10天的数据作为网络的测试样本，验证网络能否准确预测出当天的负荷数据。

根据以上分析，建立的MATLAB代码如下所示。

```
clear all
clc
% close all
% nntwarn off;
%导入题中所给样本数据
a=[0.37 0.51 0.71;...
   0.12 0.17 0.88;...
   0.32 0.99 0.69;...
   0.13 0.55 0.63;...
   0.11 0.42 0.84;...
   0.24 0.53 0.71;...
```

```
    0.33 0.44 0.9;...
    0.19 0.66 0.44;...
    0.31 0.67 0.49;...
    0.37 0.51 0.71];
%%%%%%%%%选取训练数据和测试数据%%%%%%%%%%%%
for i=1:7
    p(i,:)=[a(i,:),a(i+1,:),a(i+2,:)];
end
% 训练数据输入
p_train=p(1:6,:);
% 训练数据输出
t_train=a(4:9,:);
% 测试数据输入
p_test=p(7,:);
% 测试数据输出
t_test=a(10,:);
%为适应网络结构,对数据做转置处理
p_train=p_train';
t_train=t_train';
p_test=p_test';
%%%%%%%%%%网络的建立和训练%%%%%%%%%%%%%%%
%设置不同的隐含层神经元个数
nn=[5 10 15 20];
for i=1:4
    threshold=[0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1];
```

```
    %建立Elman神经网络隐含层为nn(i)个神经元
    net=newelm(threshold,[nn(i),3],{'tansig','purelin'});
    %设置网络训练参数
    net.trainparam.epochs=2000;
    net.trainparam.show=200;
    %初始化网络
    net=init(net);
    %Elman网络训练
    net=train(net,p_train,t_train);
    % 预测数据
    y=sim(net,p_test);
    % 计算绝对误差
    error(i,:)=y'-t_test;
end
%绘制网络在不同隐含层神经元个数时的预测绝对误差
figure(1)
plot(1:3,error(1,:),'-ro','linewidth',2);
hold on;
plot(1:3,error(2,:),'b:x','linewidth',2);
hold on;
plot(1:3,error(3,:),'k-.s','linewidth',2);
hold on;
plot(1:3,error(4,:),'c--d','linewidth',2);
title('Elman预测误差图')
set(gca,'Xtick',[1:3])
legend('5','10','15','20','location','best')
xlabel('预测时间点')
```

```
ylabel('神经元个数不同的网络预测的绝对误差')
hold off;
%计算相对误差
for i=1:4
    average(i)=(error(i,1)+error(i,2)+error(i,3))/3;
end
%绘制不同神经元个数时网络预测一天3个时间点的平均误差
figure(2)
plot(5:5:20,average,'k-.s','linewidth',2)
xlabel('建立网络的神经元个数')
ylabel('不同神经元网络预测的平均误差')
hold off;
```

运行上述代码，得到训练网络的误差曲线如图9-36所示。从图中可以看出，经过大约950步的训练，网络均方误差达到10^{-8}数量级。

神经网络在不同隐含层神经元个数时的预测绝对误差曲线图如图9-37所示。

由图9-37可以看出，神经网络预测的绝对误差均在神经元个数为5的时候，误差变化最小。不同神经元网络预测的平均误差曲线图如图9-38所示。从该图中可以清楚看出，当神经元个数为5时，网络的预测结果最好。

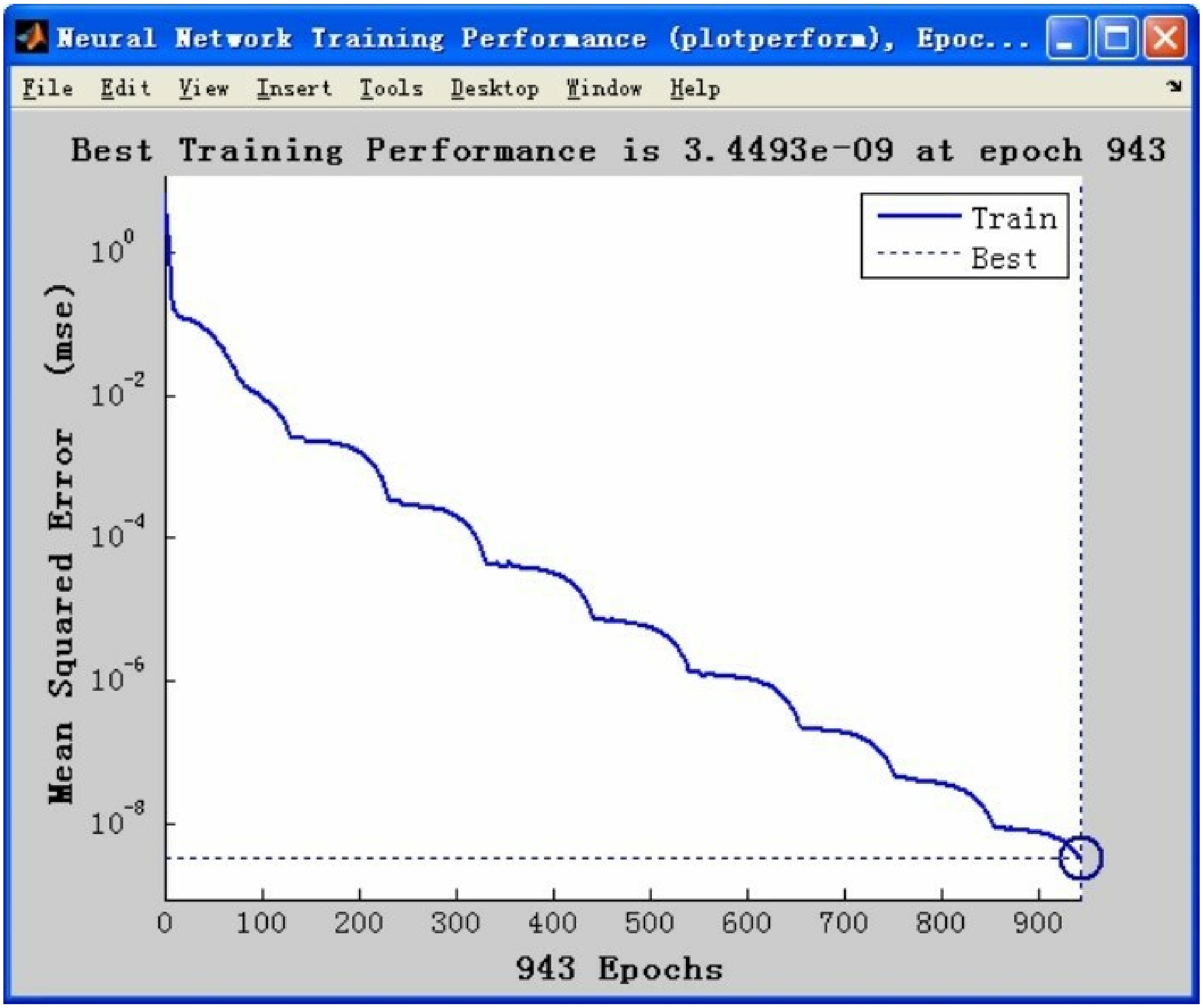


图9-36 训练网络的误差曲线

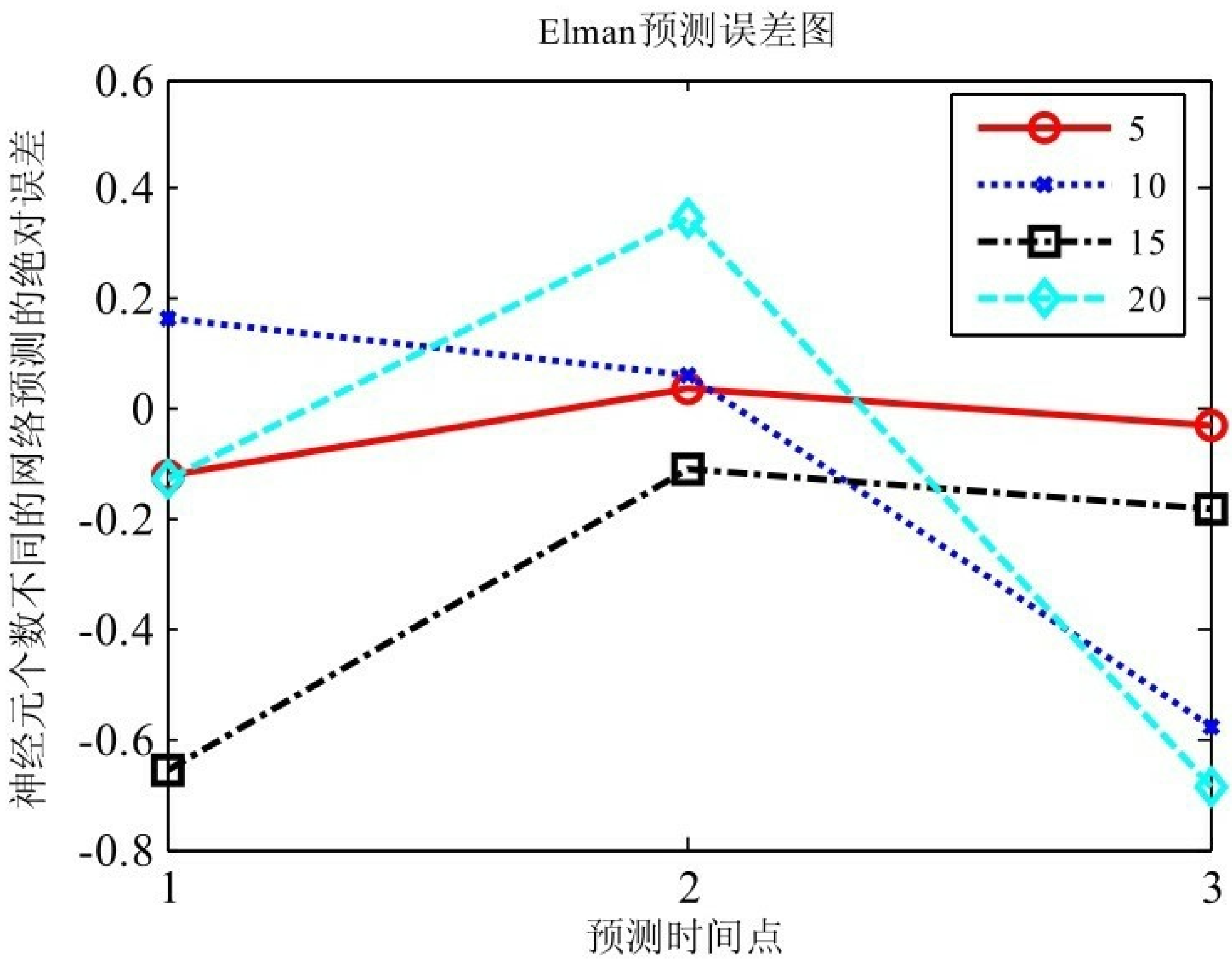


图9-37 神经元个数不同的网络预测的绝对误差曲线图

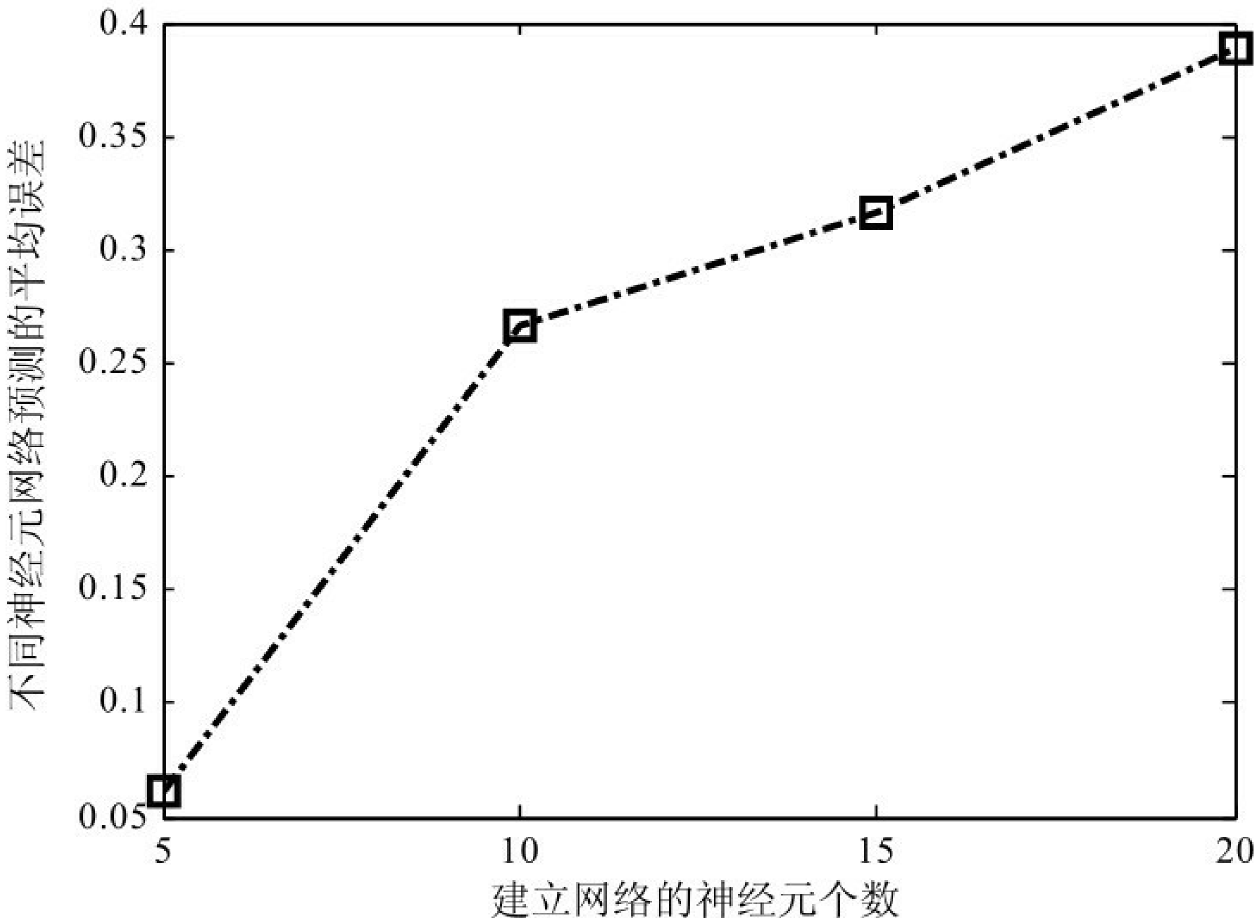


图9-38 不同神经元网络预测的平均误差曲线图

9.7 本章小结

本章介绍了反馈型神经网络，主要包括Hopfield网络和Elman网络。Hopfield网络分为离散型和连续型，本章在介绍Hopfield网络的模型和工具箱函数后，举例详细说明了离散型和连续型Hopfield网络的应用。对于Elman网络，本章对其结构、创建函数做了介绍，并举例说明了其工程应用。

第10章 竞争型神经网络

竞争型神经网络是基于无监督学习方法的神经网络的一种重要类型，它经常作为基本的网络形式，构成其他一些具有自组织能力的网络，如自组织竞争型神经网络、自组织映射网络、学习向量量化网络等。

竞争型神经网络工作的基本思想是让竞争层各神经元来竞争对输入模式相应的机会，最后仅有一个神经元成为竞争的胜者。这一获胜神经元的输出则代表对输入模式的分类。

本章将主要介绍RBF神经网络模型、工具箱函数及MATLAB应用实例。

学习目标：

·熟悉自组织神经网络模型

·掌握自适应共振理论神经网络使用方法

·了解向量量化神经网络

·掌握竞争型神经网络工具箱函数

·掌握竞争型神经网络的应用方法

10.1 自组织型竞争神经网络

在实际的神经网络中，比如人的视网膜中，存在着一种“侧抑制”现象，即一个神经细胞兴奋后，通过它的分支会对周围其他神经细胞产生抑制。

这种侧抑制使神经细胞之间出现竞争，虽然开始阶段各个神经细胞都处于程度不同的兴奋状态，由于侧抑制的作用，各细胞之间相互竞争的最终结果是，兴奋作用最强的神经细胞所产生的抑制作用战胜了它周围所有其他细胞的抑制作用而“赢”了，其周围的其他神经细胞则全“输”了。

自组织竞争人工神经网络正是基于上述生物结构和现象形成的。它能够对输入模式进行自组织训练和判断，并将其最终分为不同的类型。

与 BP 网络相比，这种自组织自适应的学习能力进一步拓宽了人工神经网络在模式识别、分类方面的应用，另一方面，竞争学习网络的核心——竞争层，又是许多种其他神经网络模型的重要组成部分。

10.1.1 几种联想学习规则

格劳斯贝格（S.Grossberg）提出了两种类型的神经元模型：内星与外星，用以解释人类及动物的学习现象。一个内星可以被训练来识别一个矢量；而外星可以被训练来产生矢量。

由r个输入构成的格劳斯贝格内星模型如图10-1所示。

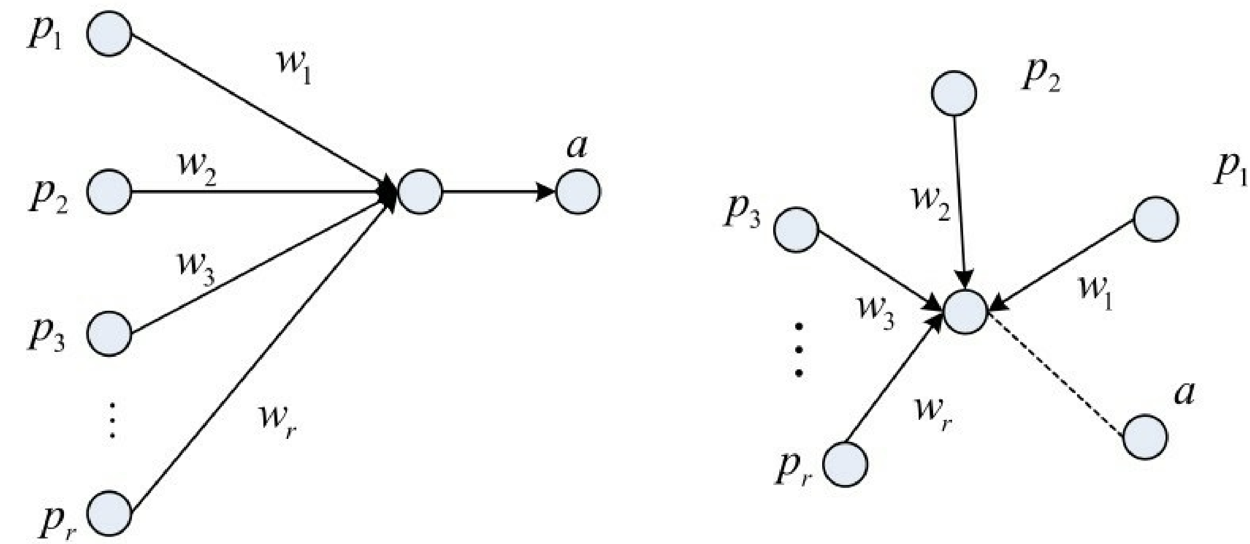


图10-1 格劳斯贝格内星模型图

由r个输出节点构成的格劳斯贝格外星模型如图10-2所示。

从图10-1和图10-2中可以清楚地看出，内星是通过联接权矢量W接受一组输入信号P；而外星则是通过联接权矢量向外输出一组信号A。

它们之所以被称为内星和外星，主要是因为其网络的结构像星形，且内星的信号流向星的内部；而外星的信号流向星的外部。下面分别详细讨论两种神经元模型的学习规则及其功效。

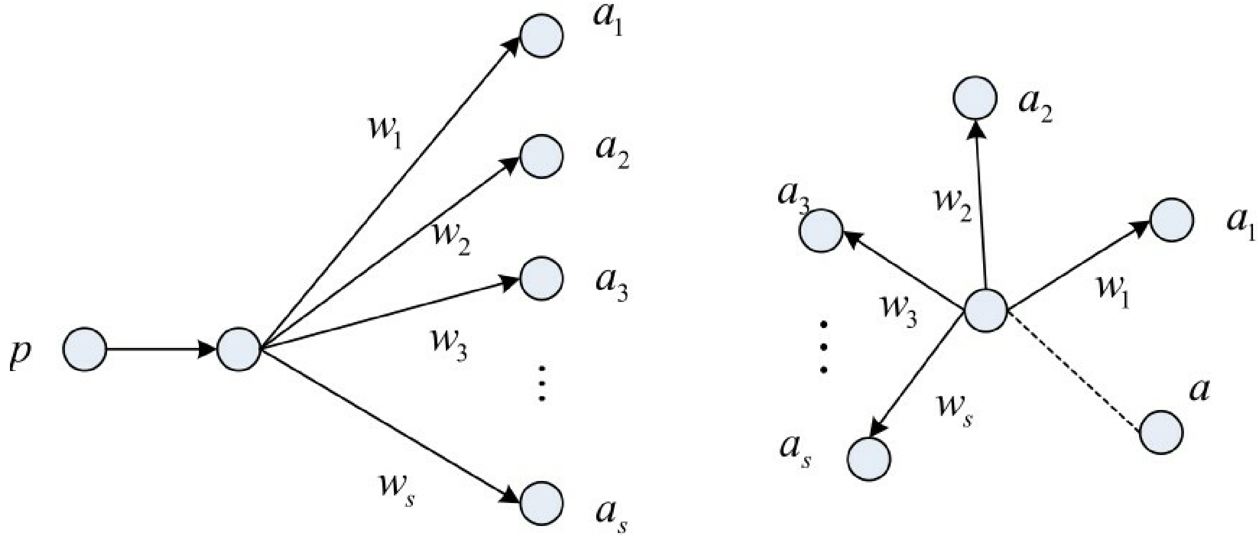


图10-2 格劳斯贝格外星模型图

1.内星学习规则

实现内星输入/输出转换的激活函数是硬限制函数。可以通过内星及其学习规则来训练某一神经元节点只响应特定的输入矢量P，它是借助于调节网络权矢量W近似于输入矢量P来实现的。

在图10-1所示的单内星中对权值修正的格劳斯贝格内星学习规则为

$$\Delta w_{1j} = \mathrm{lr} \cdot (p_j - w_{1j}) \cdot a, \ \ j = 1, 2, \cdots r$$

由上式可见，内星神经元连接强度的变化Δw_{1j}是与输出成正比的。如果内星输出a被某一外部方式维护高值时，那么通过不断反复地学习，权值将能够逐渐趋近于输入矢量 p_j的值，并趋使Δw_{1j}逐渐减少，直至最终达到$w_{1j}=p_j$，从而使内星权矢量学习了输入矢量P，达到了用内星来识别一个矢量的目的。

另一方面，如果内星输出保持为低值时，网络权矢量被学习的可能性较小，甚至不能被学习。

现在来考虑当不同的输入矢量p_1和p_2分别出现在同一内星时的情况。

首先，为了训练的需要，必须将每一输入矢量都进行单位归一化处理，即对每一个输 $1\Big/\sqrt{\sum_{j=1}^{r}(p_j^q)^2}$ 入矢量 p_q（q＝1，2），用去乘以每一个输入元素，因此所得的用来进行网络训练的新输入矢量具有单位1的模值。

当第一个矢量p_1输入给内星后，网络经过训练，最终达到$W=(p_1)^T$。此后，给内星输入另一个输入矢量p_2，此时内星的加权输入和为新矢量p_2与已学习过矢量p_1的点积，即

$$N = W \cdot P_2 = (P_1)^T \cdot P_2 = \|P_1\| \|P_2\| \cos\theta_{12} = \cos\theta_{12}$$

因为输入矢量的模已被单位化为1，所以内星的加权输入和等于输入矢量p_1和p_2之间夹角的余弦。

根据不同的情况，内星的加权输入和可分为如下几种情况。

（1）p_2等于p_1，即有θ12＝0，此时，内星加权输入和为1。

（2）p_2不等于 p_1，随着 p_2向着 p_1离开方向的移动，内星加权输入和将逐渐减少直到p_2与p_1垂直，即θ_{12}＝90°时，内星加权输入和为0。

（3）当p_2＝−p_1，即θ_{12}＝180°时，内星加权输入和达到最小值−1。

由此可见，对于一个已训练过的内星网络，当输入端再次出现该学习过的输入矢量时，内星产生1的加权输入和；而与学习过的矢量不相同的输入出现时，所产生的加权输入和总是小于1。

如果将内星的加权输入和送入到一个具有略大于−1偏差的二值型激活函数时，对于一个已学习过或接近于已学习过的矢量输入时，同样能够使内星的输出为 1，而其他情况下的输出均为0。

所以在求内星加权输入和公式中的权值W与输入矢量P的点积，反映了输入矢量与网络权矢量之间的相似度，当相似度接近l时，表明输入矢量P与权矢量相似，并通过进一步学习，能够使权矢量对其输入矢量具有更大的相似度，当多个相似输入矢量输入内星，最终的训练结果是使网络的权矢量趋向于相似输入矢量的平均值。

内星网络中的相似度由偏差 b 来控制，由设计者在训练前选定，典型的相似度值为b＝−0.95，这意味着输入矢量与权矢量之间的夹角小于18°48'。若选b＝−0.9时，则其夹角扩大为25°48'。

一层具有s个神经元的内星，可以用相似的方式进行训练，权值修正公式为

$$\Delta w_{ij} = \mathrm{lr} \cdot (p_j - w_{ij}) \cdot a$$

MATLAB 神经网络工具箱中内星学习规则的执行是用函数 learnis 来完成上述权矢量的修正过程。

dW＝1earnis(W,P,A,lr)；

W=W+dW；

一层s个内星神经元可以作为一个r到1的解码器。另外，内星通常被嵌在具有外星或其他元件构成的大规模网络中以达到某种特殊的目的。

下面给出有关训练内星网络的例子。

【例10-1】设计内星网络进行以下矢量的分类辨识。

P=[0.2 0.6;0.4 0.3;0.5 0.3;0.7 0.6];

T=[1 0];

解：与感知器分类功能不同，内星是根据期望输出值，在例题中是通过迫使网络在第一个输入矢量出现时，输出为1，同时迫使网络在第二个输入矢量出现时，输出为0，而使网络的权矢量逼近期望输出为1的第一个输入矢量。

首先对网络进行初始化处理。

```
clear all
clc
P=[0.2 0.6;0.4 0.3;0.5 0.3;0.7 0.6];
T=[1 0];
[R,Q]=size(P);
[S,Q]=size(T);
W=zeros(S,R);
max_epoch=10;
lp.lr = 0.2;
```

注意权矢量在此是进行了零初始化，这里的学习速率的选择也具有任意性，当输入矢量较少时，学习速率可以选择得较大以加快学习收敛速度。另外，因为所给例题中所给输入矢量已是归一化后的值，所以不用再作处理。下面是设计训练内星网络的程序。

```
for epoch=1:(max_epoch)
   for q=1:Q
      A=T(q);
      dW=learnis(W,P(:,q),[],[],A,[],[],[],[],[],lp,[]);
      W=W+dW;
   end
end
```

经过10次循环以及480次计算后，得到的权矢量为

W＝[0.1785　0.3571　0.4463　0.6248]

而当lr=0.5时，其结果为

W=[0.1998　0.3996　0.4995　0.6993]

由此可见，学习速率较低时，在相同循环次数下，其学习精度较低。但当输入矢量较多时，较大的学习速度可能产生波动，所以要根据具体情况来确定参数值。

2.外星学习规则

外星网络的激活函数是线性函数，它被用来学习回忆一个矢量，其网络输入P也可以是另一个神经元模型的输出。外星被训练来在一层 s 个线性神经元的输出端产生一个特别的矢量A。所采用的方法与内星识别矢量时的方法极其相似。

对于一个外星，其学习规则为

$$\Delta w_{i1} = \mathrm{lr} \cdot (a_i - w_{i1}) \cdot p_j$$

与内星不同，外星连接强度的变化Δw是与输入矢量P成正比的。这意味着当输入矢量被保持高值，比如接近1时，每个权值w_{ij}将趋于输出a_i值，若p_j＝1，则外星使权值产生输出矢量。

当输入矢量p_j为0时，网络权值得不到任何学习与修正。

当有r个外星相并联，每个外星与s个线性神经元相连组成一层外星

时，每当某个外星的输入节点被置为1时，与其相连的权值到矢量w_{ij}就会被训练成对应的线性神经元的输出矢量A，其权值修正方式为

$$\Delta W = \mathrm{lr} \cdot (A - W) \cdot P$$

其中：W＝s×r权值列矢量；lr＝学习速率；A＝s×q外星输出；P＝r×q外星输入。

MATLAB工具箱中实现外星学习与设计的函数为learnos.m，其调用过程如下。

dW＝learnos(W,A,P,lr)；

W＝W+dW；

【例 10-2】给定两元素的输入矢量以及与它们相关的四元素目标矢量，设计一个外星网络实现有效的矢量的获得，外星没有偏差。

```
T=[0.2 0.6;0.4 0.3;0.5 0.3;0.7 0.6];
P=[1 0];
```

解：该网络的每个目标矢量强迫为网络的输出，而输入只有 0 或 1。网络训练的结果是使其权矩阵趋于所对应的输入为1时的目标矢量。

同样网络被零权值初始化。

```
clear all
clc
T=[0.2 0.6;0.4 0.3;0.5 0.3;0.7 0.6];
P=[1 0];
[R,Q]=size(P);
[S,Q]=size(T);
W=zeros(S,R);
max_epoch=10;
lp.lr = 0.3;
```

下面根据外星学习规则进行训练。

```
for epoch=1:(max_epoch)
   for q=1:Q
      A=T(:,q);
      dW=learnos(W,P(:,q),[],[],A,[],[],[],[],[],lp,[]);
      W=W+dW;
   end
end
W
```

一旦训练完成，当外星工作时，对设置于输入为1的矢量，将能够回忆起被记忆在网络中的第一个目标矢量的近似值。

```
Ptest＝[1]；
A＝pureline(W*Ptest)
A＝
   0.1944
   0.3887
   0.4859
   0.6802
```

由此可见，此外星已被学习来回忆起了第一个矢量。事实上，它被学习来回忆出在【例10-1】中学习识别出的那个矢量。即上述外星的权值非常接近于【例 10-1】中已被识别的矢量。

内星与外星之间的对称性是非常有用的。对一组输入和目标来训练一个内星层与将其输入与目标相对换，来训练一个外星层的结果是相同的，即它们之间的权矩阵的结果是相互转置的。

3.科荷伦学习规则

科荷伦学习规则是由内星规则发展而来的。对于其值为0或1的内星输出，当只对输出为1的内星权矩阵进行修正，即学习规则只应用于输出为1的内星上，将内星学习规则中的a_i取值1，则可导出科荷伦规则为

$$\Delta w_{ij} = \text{lr} \cdot (p_j - w_{ij})$$

科荷伦学习规则实际上是内星学习规则的一个特例，但它比采用内星规则进行网络设计要节省更多的学习，因而常常用来替代内星学习规则。

在 MATLAB 工具箱中，在调用科荷伦学习规则函数 learnk.m 时，一般通过先寻找输出为1的行矢量i，然后仅对与i相连的权矩阵进行修正。使用方法如下。

```
i＝find(A==1)；
dW＝learnk(W，P，i，1r)；
W＝W+dW；
```

一般情况下科荷伦学习规则比内星学习规则能够提高训练速度1～2个数量级。

10.1.2 网络结构

竞争网络由单层神经元网络组成，其输入节点与输出节点之间为全互联结。因为网络在学习中的竞争特性也表现在输出层上，所以在竞争网络中把输出层又称为竞争层，而与输入节点相连的权值及其输入合称为输入层。实际上，在竞争网络中，输入层和竞争层的加权输入和共用同一个激活函数，如图10-3所示。

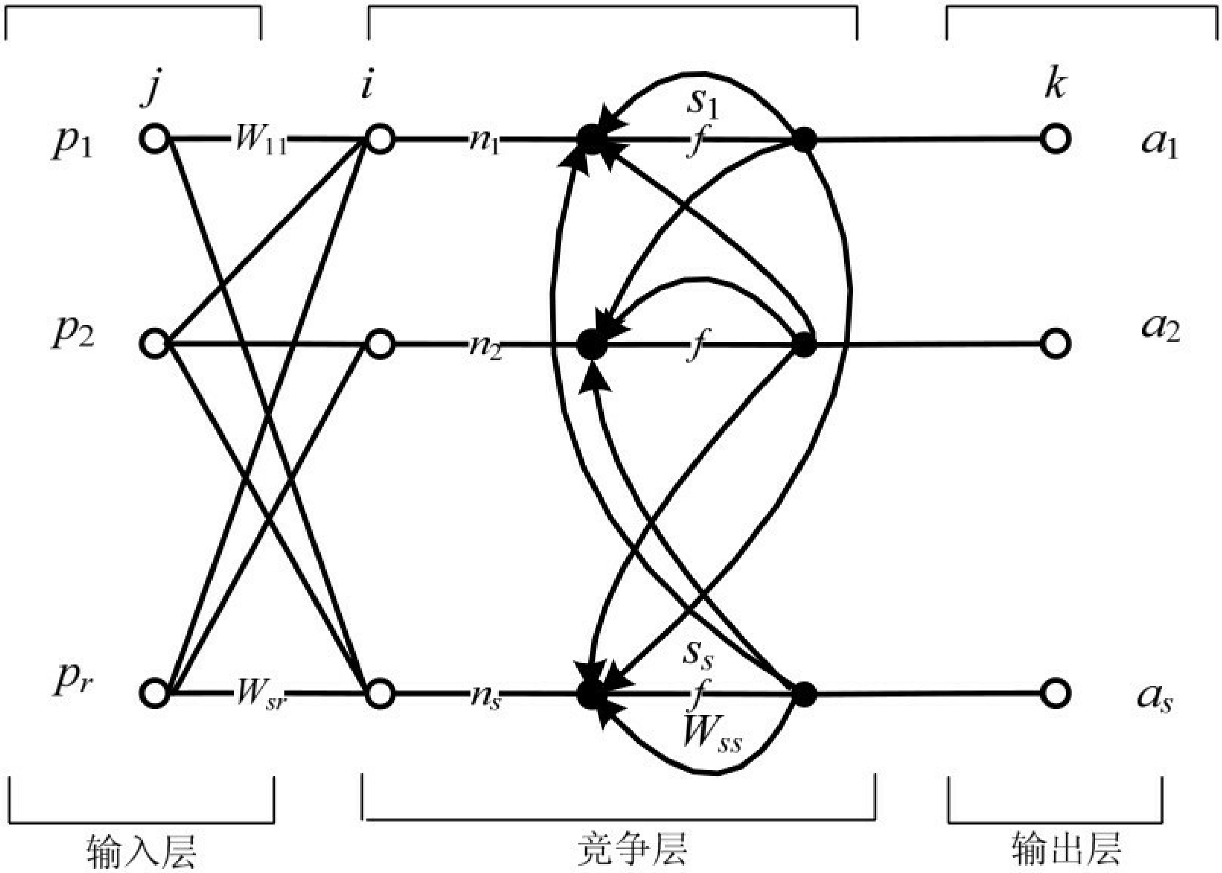


图10-3 竞争网络结构图

从网络的结构图中可以看出，自组织竞争网络的权值有两类：一类是输入节点 j 到 i的权值 w_{ij}(i＝1,2,…,s；j＝1,2,…,r)，这些权值是通过训练可以被调整的；另一类是竞争层中互相抑制的权值w_{ik}(k＝1,2,…,s)。这类权值是固定不变的，且它满足一定的分布关系，如竞争网络的激活函数为二值型{0，1}函数。

距离相近的抑制强，距离远的抑制弱。

另外，它们是一种对称权值，即有$w_{ik}=w_{ki}$，同时相同神经元之间的权值起加强的作用，即满足$w_{11}=w_{11}=\ldots=w_{kk}>0$，而不同神经元之间的权值相互抑制，对于$k\neq i$有$w_{ij}<0$。

下面来具体分析竞争网络的输出情况。

设网络的输入矢量为：$P=[p_1，p_2\ldots p_r]^T$；

对应网络的输出矢量为：$A=[a_1，a_2\ldots a_s]^T$。

由于竞争网络中含有两种权值，所以其激活函数的加权输入和也分为两部分：来自输入节点的加权输入和N与来自竞争层内互相抑制的加权输入和G。具体地说，对于第i个神经元有如下情况。

（1）来自输入节点的加权输入和为

$$n_i = \sum_{j=1}^{r} w_{ij} \cdot p_j$$

（2）来自竞争层内互相抑制的加权输入和为

$$g_i = \sum_{k \in D} w_{ik} \cdot f(a_k)$$

这里D表示竞争层中含有神经元节点的某个区域，如果D表示的是整个竞争层，竞争后只能有一个神经元兴奋而获胜；如果竞争层被分成若干个区域，则竞争后每个区域可产生一个获胜者。

由于g_i与网络的输出值a_k有关，而输出值又是由网络竞争后的结果所决定的，所以g_i的值也是由竞争结果确定的。为了方便起见，下面以D为整个网络输出节点的情况来分析竞争层内互相抑制的加权输入和g_i的可能结果。

（a）如果在竞争后，第i个节点“赢”了，则有

$$a_k = 1, k = i$$

而其用户所有节点的输出均为零，即

$$a_k = 0, k = 1, 2, \cdots, s; k \neq i$$

$$g_i = \sum_{k=1}^{s} w_{ik} \cdot f(a_k) = w_{ii} > 0$$

此时，

（b）如果在竞争后，第i个节点“输”了，而“赢”的节点为L，则有

$$a_k = 1, k = L$$

$$a_k = 0, k = 1, 2, \cdots, s; k \neq L$$

$$g_i = \sum_{k=1}^{s} w_{ik} \cdot f(a_k) = w_{iL} < 0$$

此时，

所以对整个网络的加权输入总和有下式成立。

$S_i = n_i + w_{iL}$ 对于“赢”的节点L

$S_i = n_i - |w_{ii}|$ 对于所有“输”的节点

$i = 1, 2, \cdots, s; i \neq L$

由此可以看出，经过竞争后只有获胜的那个节点的加权输入总和为最大。竞争网络的输出为

$$a_k = \begin{cases} 1 & (s_k = \max(s_i, i=1,2,\cdots,s)) \\ 0 & \text{其他} \end{cases}$$

因为在权值的修正过程中只修正输入层中的权值 w_{ij}，竞争层内的权值 w_{ik}是固定不变的，它们对改善竞争的结果只起到了加强或削弱的作用，即对获胜节点增加一个正值，使其更易获胜，对输出的节点增加一个负值，使其更不易获胜，而对改变节点竞争结果起决定性作用的还是输入层的加权和n_i，所以在判断竞争网络节点胜负的结果时，可直接采用n_i，即

$$n_{赢}=\max(\sum_{j=1}^{r} w_{ij} p_j)$$

取偏差B为零是判定竞争网络获胜节点时的典型情况，偶而也采用下式进行竞争结果的判定

$$n_{赢}=\max(\sum_{j=1}^{r} w_{ij} p_j + b), -1 < b < 0$$

典型的b值取−0.95。加上b值意味着取b＝−$|w_{i1}|$这一最坏的情况。

通过上面分析，可以将竞争网络的工作原理总结如下：竞争网络的激活函数使加权输入和为最大的节点赢得输出为 1，而其用户神经元的输出皆为 0。这个竞争过程可用MATLAB描述如下。

```
n＝W*P；
[S，Q]＝size(n)；
x＝n+b*ones(1，Q)；
y＝max(x)；
for q＝1：Q
s＝find(x(:, q)＝y(q))；    %找出最大加权输入和y(q)所在的行；
a(z(1)，q)＝1；        %令元素 a(z，q)＝1，其用户值为零；
```

这个竞争过程的程序已被包含在竞争激活函数compet.m之中，可与其用户函数一样简单的方式来调用它即可得到竞争网络的输出值。

```
A＝compet(W*P，B)；
```

10.1.3 自组织神经网络的原理

分类是在类别知识等导师信号的指导下，将待识别的输入模式分配

到各自的模式类中，无导师指导的分类称为聚类，聚类的目的是将相似的模式样本划归一类，而将不相似的分离开来，实现模式样本的类内相似性和类间分离性。

由于无导师学习的训练样本中不含期望输出，因此对于某一输入模式样本应属于哪一类并没有任何先验知识。对于一组输入模式，只能根据它们之间的相似程度来分为若干类，因此，相似性测量是输入模式的聚类依据。

1.相似性测量

神经网络的输入模式向量的相似性测量可用向量之间的距离来衡量。常用的方法有欧氏距离法和余弦法两种。

（1）欧式距离法。

设X,X_i为两向量，其间的欧式距离为

$$d=\left\|X-X_i\right\|=\sqrt{(X-X_i)(X-X_i)^T}$$

d越小， X 与X_i越接近，两者越相似，当d=0时， X =X_i；以d=T（常数）为判据，可对输入向量模式进行聚类分析。

由于d_{12},d_{23},d_{31}均小于T， d_{45},d_{56},d_{46}均小于T，而

$d_{1i}>T(i=4,5,6)$,

$d_{2i}>T(i=4,5,6)$ ， $d_{3i}>T(i=4,5,6)$，故将输入模式X_1, X_2, X_3, X_4, X_5, X_6分为类1和类2两大类，如图10-4所示。

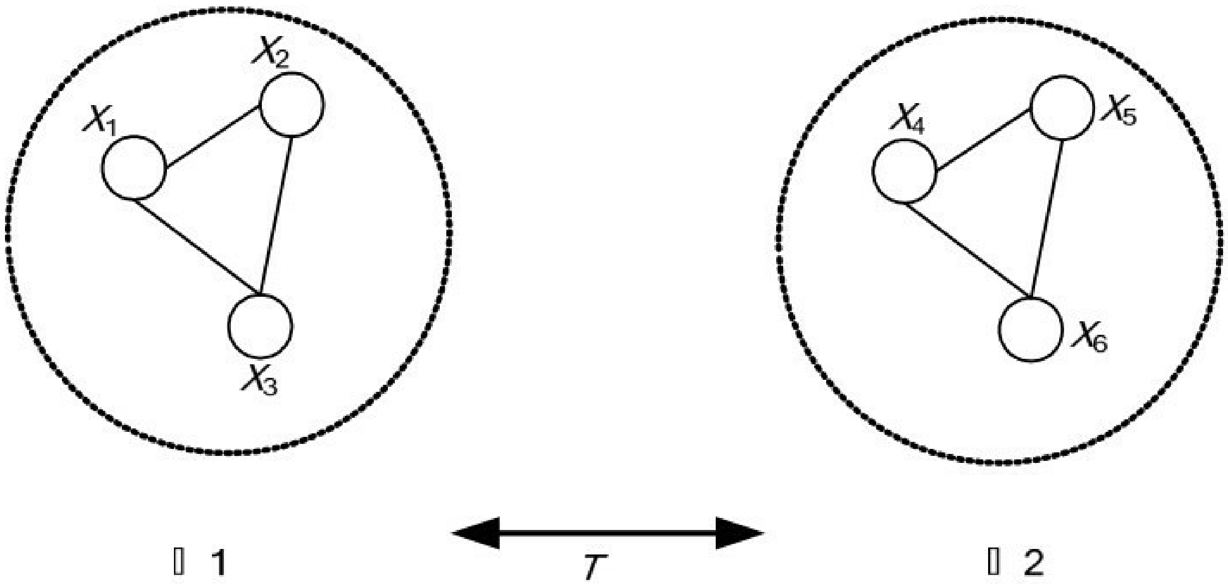


图10-4 基于欧式距离法的模式分类

（2）余弦法。

设X,X_i为两向量，其间的夹角余弦

$$\cos\phi = \frac{XX^T}{\|X\|\|X_i\|}$$

φ越小，　X　与X_i越接近，两者越相似；当φ＝0时，cosφ＝1，　X =X_i；同样以φ=φ_0为判据可进行聚类分析。

2.竞争学习原理

竞争学习规则的生理学基础是神经细胞的侧抑制现象：当一个神经细胞兴奋后，会对其周围的神经细胞产生抑制作用。最强的抑制作用是竞争获胜的“唯我独兴”，这种做法称为“胜者为王”（Winner-Take-All，WTA）。竞争学习规则就是从神经细胞的侧抑制现象获得的。它的学习步骤如下。

（1）向量归一化。

对自组织网络中的当前输入模式向量 X、竞争层中各神经元对应的内星权向量w_i(j=1,2,",m)，全部进行归一化处理，如图10-5所示，得到

$\hat{X}$和$\hat{w}_j$如下。

$$\hat{X} = \frac{X}{\|X\|}, \hat{W}_j = \frac{W_j}{\|W_j\|}$$

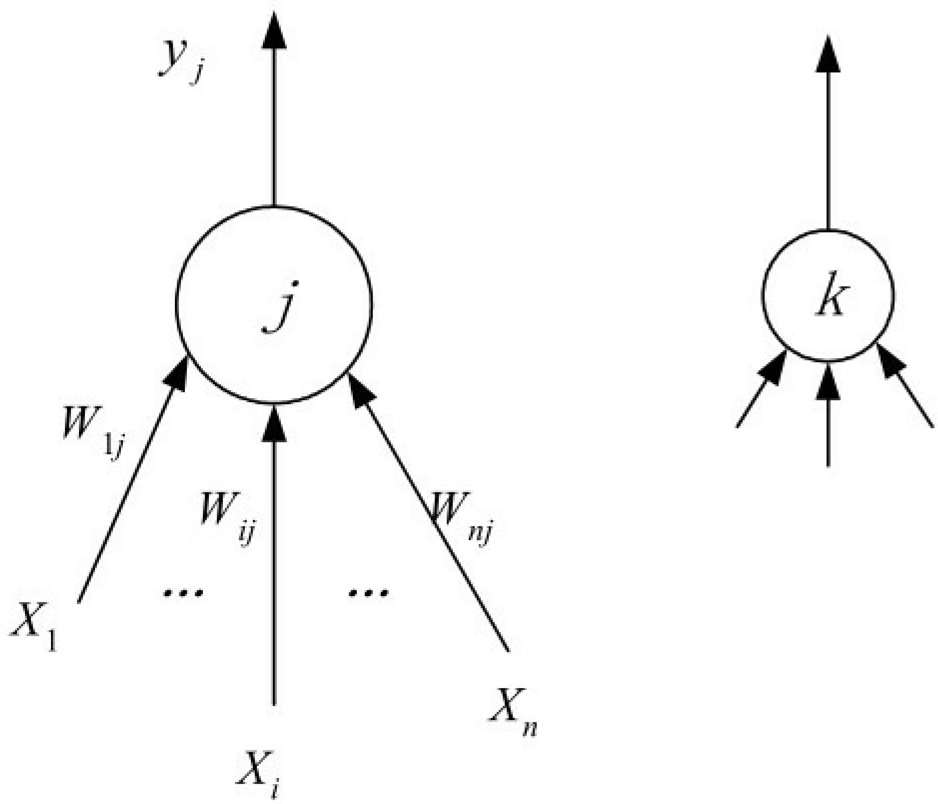


图10-5 向量归一化

（2）寻找获胜神经元。

将$\hat{X}$与竞争层所有神经元对应的内星权向量$\hat{w}_j\,(j=1,2,\cdots,m)$进行相似性比较。最相似的神经元获胜，权向量为$\hat{W}_{j^*}$。

$$\left\|\hat{X}-\hat{W}_{j^*}\right\| = \min_{j\in\{1,2,\cdots,n\}}\left\{\left\|\hat{X}-\hat{W}_j\right\|\right\}$$

$$\Rightarrow \left\|\hat{X}-\hat{W}_{j^*}\right\| = \sqrt{(\hat{X}-W_{j^*})(\hat{X}-W_{j^*})^T} = \sqrt{\hat{X}\hat{X}^T - 2\hat{W}_{j^*}\hat{X}^T + \hat{W}_{j^*}\hat{W}_{j^*}{}^T} = \sqrt{2(1-\hat{W}_{j^*}\hat{X}^T)}$$

$$\Rightarrow \hat{W}_{j^*}\hat{X}^T = \max_j(\hat{W}_j\hat{X}^T)$$

（3）网络输出与权调整

按WTA学习法则，获胜神经元输出为“1”，其余为0。即

$$y_j(t+1) = \begin{cases} 1 & j = j^* \\ 0 & j \neq j^* \end{cases}$$

只有获胜神经元才有权调整其权向量 W_{j^*}。其权向量学习调整如下。

$$\begin{cases} W_{j^*}(t+1) = \hat{W}_{j^*}(t) + \Delta W_{j^*} = \hat{W}_{j^*}(t) + \alpha(\hat{X} - \hat{W}_{j^*}) \\ W_j(t+1) = \hat{W}_j(t) & j \neq j^* \end{cases}$$

$0<\alpha\leq 1$为学习率，α一般随着学习的进展而减小，即调整的程度越来越小，趋于聚类中心。

（4）重新归一化处理

归一化后的权向量经过调整后，得到的新向量不再是单位向量，因此要对学习调整后的向量重新进行归一化，循环运算，直到学习率α衰减到0。

为了更好地说明竞争学习算法的聚类分析效果，下面以一具体实例进行计算。

【例10-3】用竞争学习算法将下列各模式分为两类。

$X_1 = (0.8,0.6)$， $X_2 = (0.1736,-0.9848)$， $X_3 = (0.707,0.707)$，

$X_4 = (0.342, -0.9397)$， $X_5=(0.6,0.8)$，学习率α＝0.5。

解：将上述输入模式转换为极坐标形式：

$X_1 = 1\angle 36.89^\circ$， $X_2 = 1\angle -80^\circ$， $X_3 = 1\angle 45^\circ$， $X_4 = 1\angle -70^\circ$， $X_5 = 1\angle 53.13^\circ$。如图10-6所示。

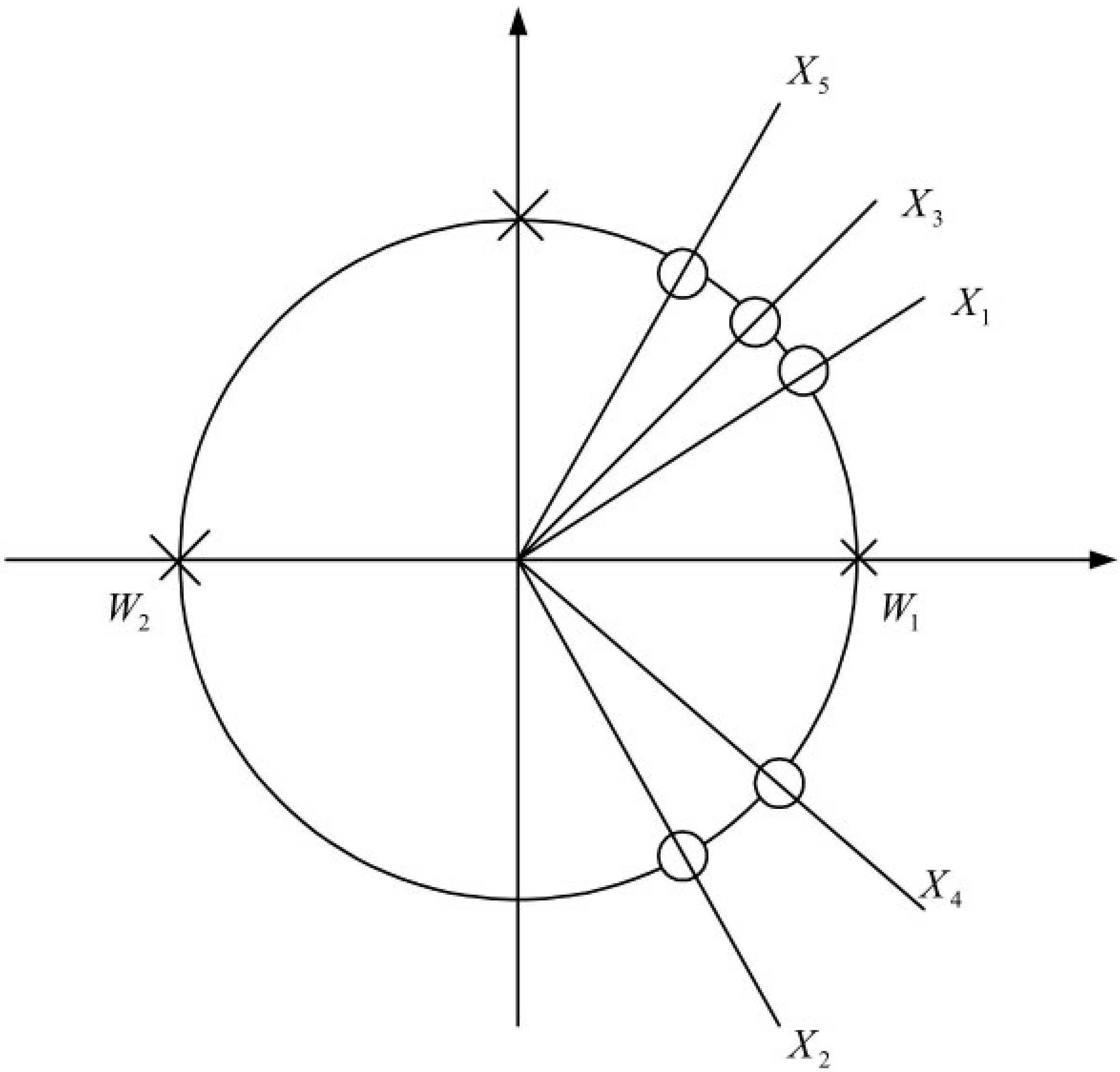


图10-6 模式向量图

要求将各模式分为两类，则竞争层为两个神经元，设两个权向量，

随机初始化为单元向量：

$$W_1(0)=(1,0)=1\angle 0^\circ\text{，}\quad W_2(0)=(-1,0)=1\angle -180^\circ$$

，其竞争学习过程如下。

（1） X_1。

$$d_1=\|X_1-W_1(0)\|=1\angle 36.89^\circ\text{，}\quad d_2=\|X_1-W_2(0)\|=1\angle 216.89^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(1)=W_1(0)+\alpha(X_1-W_1(0))=0+0.5\times 36.89=1\angle 18.43^\circ$$

$$W_2(1)=W_2(0)=1\angle -180^\circ$$

（2） X_2。

$$d_1=\|X_2-W_1(1)\|=1\angle 98.43^\circ\text{，}\quad d_2=\|X_2-W_2(1)\|=1\angle 100^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(2)=W_1(1)+\alpha(X_2-W_1(1))=18.43+0.5\times(-80-18.43)=1\angle -30.8^\circ$$

$$W_2(2)=W_2(1)=1\angle -180^\circ$$

（3） X_3。

$$d_1=\|X_3-W_1(2)\|=1\angle 75.8^\circ\text{，}\quad d_2=\|X_3-W_2(2)\|=1\angle 225^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(3)=W_1(2)+\alpha(X_3-W_1(2))=-30.8+0.5\times(45+30.8)=1\angle 7^\circ$$

$$W_2(3)=W_2(2)=1\angle -180^\circ$$

（4） X_4。

$$d_1=\|X_4-W_1(3)\|=1\angle 77^\circ\text{，}\quad d_2=\|X_4-W_2(3)\|=1\angle 110^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(4)=W_1(3)+\alpha(X_4-W_1(3))=7+0.5\times(-70-7)=1\angle -31.5^\circ$$

$$W_2(4)=W_2(3)=1\angle -180^\circ$$

（5） X_5。

$$d_1 = \|X_5 - W_1(4)\| = 1\angle 84.63^\circ，\quad d_2 = \|X_5 - W_2(4)\| = 1\angle 126.87^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(5) = W_1(4) + \alpha(X_5 - W_1(4)) = -31.5 + 0.5\times(53.13 + 31.5) \approx 1\angle 11^\circ$$

$$W_2(5) = W_2(4) = 1\angle -180^\circ$$

（6） X_1。

$$d_1 = \|X_1 - W_1(5)\| = 1\angle 25.89^\circ，\quad d_2 = \|X_1 - W_2(5)\| = 1\angle 216.89^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(6) = W_1(5) + \alpha(X_1 - W_1(5)) = 11 + 0.5\times 25.89 \approx 1\angle 24^\circ$$

$$W_2(6) = W_2(5) = 1\angle -180^\circ$$

（7） X_2。

$$d_1 = \|X_2 - W_1(6)\| = 1\angle 104^\circ，\quad d_2 = \|X_2 - W_2(6)\| = 1\angle 100^\circ$$

$d_2<d_1$，神经元2获胜，W_2调整。

$$W_2(7) = W_2(6) + \alpha(X_2 - W_2(6)) = -180 + 0.5\times(-80 + 180) = 1\angle -130^\circ$$

$$W_1(7) = W_1(6) = 1\angle 24^\circ$$

（8） X_3。

$$d_1 = \|X_3 - W_1(7)\| = 1\angle 21^\circ，\quad d_2 = \|X_3 - W_2(7)\| = 1\angle 175^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(8) = W_1(7) + \alpha(X_3 - W_1(7)) = 24 + 0.5\times(45 - 24) \approx 1\angle 34^\circ$$

$$W_2(8) = W_2(7) = 1\angle -130^\circ$$

（9） X_4。

$$d_1 = \|X_4 - W_1(8)\| = 1\angle 104^\circ，\quad d_2 = \|X_4 - W_2(8)\| = 1\angle 60^\circ$$

$d_2<d_1$，神经元2获胜，W_2调整。

$$W_2(9) = W_2(8) + \alpha(X_4 - W_2(8)) = -130 + 0.5\times(-70 + 130) = 1\angle -100^\circ$$

$$W_1(9)=W_1(8)=1\angle 34^\circ$$

（10） X_5。

$$d_1=\left\|X_5-W_1(9)\right\|=1\angle 19.13^\circ，\quad d_2=\left\|X_5-W_2(9)\right\|=1\angle 153.13^\circ$$

$d_1<d_2$，神经元1获胜，W_1调整。

$$W_1(10)=W_1(9)+\alpha(X_5-W_1(9))=34+0.5\times(53.13-34)\approx 1\angle 44^\circ$$

$$W_2(10)=W_2(9)=1\angle -100^\circ$$

一直循环运算下去，其前20次学习结果见表10-1。

表10-1 竞争学习结果

学习次数	W_1	W_2	学习次数	W_1	W_2
1	18.43°	−180°	11	40.5°	−100°
2	−30.8°	−180°	12	40.5°	−90°
3	7°	−180°	13	43°	−90°
4	−32°	−180°	14	43°	−81°
5	11°	−180°	15	47.5°	−81°
6	24°	−180°	16	42°	−81°
7	24°	−130°	17	42°	−80.5°
8	34°	−130°	18	43.5°	−80.5°
9	34°	−100°	19	43.5°	−75°
10	44°	−100°	20	48.5°	−75°

从表10-1可见，在运算学习20次后，网络权值W_1，W_2趋于稳定：

$W_1\to 45^\circ, W_2\to -75^\circ$。同时还可以得到如下结论。

·当W_1调整时，W_2不变，反之亦然，每次只有一个神经元获胜。

•X_1,X_3,X_5属于同一模式，其中心向量为

$$\frac{1}{3}(X_1+X_2+X_3)=45^\circ$$

；X_2，X_4属于同一模式类，为

$$\frac{1}{2}(X_2+X_4)=1\angle-75^\circ$$

。

·若学习率α为常数，W_1，W_2将在中心向量附近摆动，永远也不收敛。

10.1.4 竞争学习规则

竞争网络在经过竞争而求得获胜节点后，则对与获胜节点相连的权值进行调整，调整权值的目的是为了使权值与其输入矢量之间的差别越来越小，从而使训练后的竞争网络的权值能够代表对应输入矢量的特征，把相似的输入矢量分成了同一类，并由输出来指示所代表的类别。

竞争网络修正权值的公式为 $\Delta w_{ij}=\mathrm{lr}\cdot(p_j-w_{ij})$

式中lr为学习速率，且0＜lr＜1，一般的取值范围为0.01−0.3；p_j为经过归一化处理后的输入。

用MATLAB工具箱来实现上述公式的过程可以用内星学习规则。

A＝compet(W*P)；

dW＝learnis(P，A，lr，W)；

W＝W+dW；

更省时地是采用科荷伦学习规则，如下。

A＝compet(W*P)；

i＝find(A==1)；

dW＝learnis(P，i，lr，W)；

W＝W+dW；

不论采用哪种学习方法，层中每个最接近输入矢量的神经元，通过每次权值调整而使权值矢量逐渐趋于这些输入矢量。从而竞争网络通过

学习而识别了在网络输入端所出现的矢量，并将其分为某一类。

10.1.5 竞争网络的训练过程

理解网络的训练过程是为了更好地设计出网络。

因为只有与获胜节点相连的权值才能得到修正，通过其学习法则使修正后的权值更加接近其获胜输入矢量。结果是，获胜的节点对将来再次出现的相似矢量（能被偏差公式所包容，或在偏差范围以内的），更加容易赢得该节点的胜利。

而对于一个不同的矢量出现时，就更加不易取胜，但可能使其用户某个节点获胜，归为另一类矢量群中。随着输入矢量的重复出现而不断地调整与胜者相连的权矢量，以使其更加接近于某一类输入矢量。

最终，如果有足够的神经元节点，每一组输入矢量都能使一节点的输出为1而聚为该类。通过重复训练，自组织竞争网络将所有输入矢量进行了分类。

所以竞争网络的学习和训练过程，实际上是对输入矢量的划分聚类过程，使得获胜节点与输入矢量之间的权矢量代表获胜输入矢量。

这样，当达到最大循环的值后，网络已重复多次训练了P中的所有矢量，训练结束后，对于用于训练的模式P，其网络输出矢量中，其值为1 的代表一种类型，而每类的典型模式值由该输出节点与输入节点相连的权矢量表示。

竞争网络的输入层节点r是由已知输入矢量决定的，但竞争层的神经元数s是由设计者确定的，它们代表输入矢量可能被划为的种类数，其值若被选得过少，则会出现有些输入矢量无法被分类的不良结果。

但若被选得太大，竞争后可能有许多节点都被空闲，而且在网络竞争过程中还占用了大量的设计量和时间，在一定程度上造成了一定的浪费，所以一般情况下，可以根据输入矢量的维数及其估计，再适当地增

加些数目来确定。

另外还要事先确定的参数有学习速率和最大循环次数。竞争网络的训练是在达到最大循环次数后停止，这个数一般可取输入矢量数组的15～20倍，即使每组输入矢量能够在网络重复出现15～20次。

竞争网络的权值要进行随机归一化的初始化处理，这个过程在MATLAB 中用函数randnr实现。

w＝randnr(S，R)；

然后网络则可以进入竞争以及权值的调整阶段。

网络的训练全过程完全由计算机去做，工具箱中的竞争网络训练函数为trainc，调用时所需要的参数为：初始权矩阵W，输入矩阵P和具有3个训练参数的行矢量：显示循环的频率disp_freq，最大的训练次数max_cycle以及学习速率lr，它们的用法如下。

TP=[disp_freq max_cycle lr];

W=train(W,P,TP)

竞争网络比较适合用于具有大批相似数组的分类问题，但当遇到大量的具有概率分布的输入矢量时，竞争网络就无能为力了，这时可以采用自组织特征映射神经网络来解决。

10.2 自组织特征映射神经网络

神经细胞模型中还存在着一种细胞聚类的功能柱。它是由多个细胞聚合而成的，在接受外界刺激后，它们会自动形成。一个功能柱中的细胞完成同一种功能。

生物细胞中的这些现象在自组织特征映射神经网络模型中有所反映。当外界输入不同的样本到科荷伦自组织映射网络中，一开始时输入样本引起输出兴奋的位置各不相同，但通过网络自组织后会形成一些输出群，它们分别代表了输入样本的分布，反映了输入样本的图形分布特征，所以自组织特征映射神经网络常常被称为特性图。

自组织特征映射神经网络使输入样本通过竞争学习后，功能相同的输入靠得比较近，不同的分得比较开，以此将一些无规则的输入自动排开，在联接权的调整过程中，使权的分布与输入样本的概率密度分布相似。

自组织特征映射神经网络可以作为一种样本特征检测器，在样本排序、样本分类以及样本检测方面有广泛地应用。

一般可以这样说，自组织特征映射神经网络的权矢量收敛到所代表的输入矢量的平均值，它反映了输入数据的统计特性。再扩大一点，如果说一般的竞争学习网络能够训练识别出输入矢量的点特征，那么自组织特征映射神经网络能够表现出输入矢量在线上或平面上的分布特征。

当随机样本输入到自组织特征映射神经网络时，如果样本足够多，那么在权值分布上可近似于输入随机样本的概率密度分布，在输出神经元上也反映了这种分布，即概率大的样本集中在输出空间的某一个区域。

如果输入的样本有几种分布类型，则它们各自会根据其概率分布集

中到输出空间的各个不同的区域。每一个区域代表同一类的样本，这个区域可逐步缩少，使区域的划分越来越明显。

在这种情况下，不论输入样本是多少维的，都可投影到低维的数据空间的某个区域上。这种形式也称为数据压缩。同时，如果高维空间中比较相近的样本，则在低维空间中的投影也比较相近，这样就可以从中取出样本空间中较多的信息。

不过，网络在高维映射到低维时会畸变，且压缩比越大，畸变越大；另外，网络要求的输入节点数很大，因而自组织特征映射神经网络比其他人工神经网络（如 BP 网络）的规模要大。

10.2.1 自组织特征映射神经网络拓扑结构

自组织特征映射神经网络结构也是两层：输入层和竞争层。与基本竞争网络不同之处是其竞争层可以由一维或二维网络矩阵方式组成，且权值修正的策略也不同。

（1）一维网络结构与基本竞争学习网络相同。

（2）二维网络结构，如图10-7所示，网络上层有输出节点s个，按二维形式排成一个节点矩阵，输入节点处于下方，有r个矢量，即r个节点，所有输入节点到所有输出节点之间都有权值连接，而且在二维平面上的输出节点相互间也可能是局部连接的。

自组织特征映射神经网络的激活函数为二值型函数。一般情况下b值固定，其学习方法与普通的竞争学习算法相同。

在竞争层，每个神经元都有自己的邻域，图10-8所示为一个在二维层中的主神经元。主神经元具有在其周围增加直径的邻域。

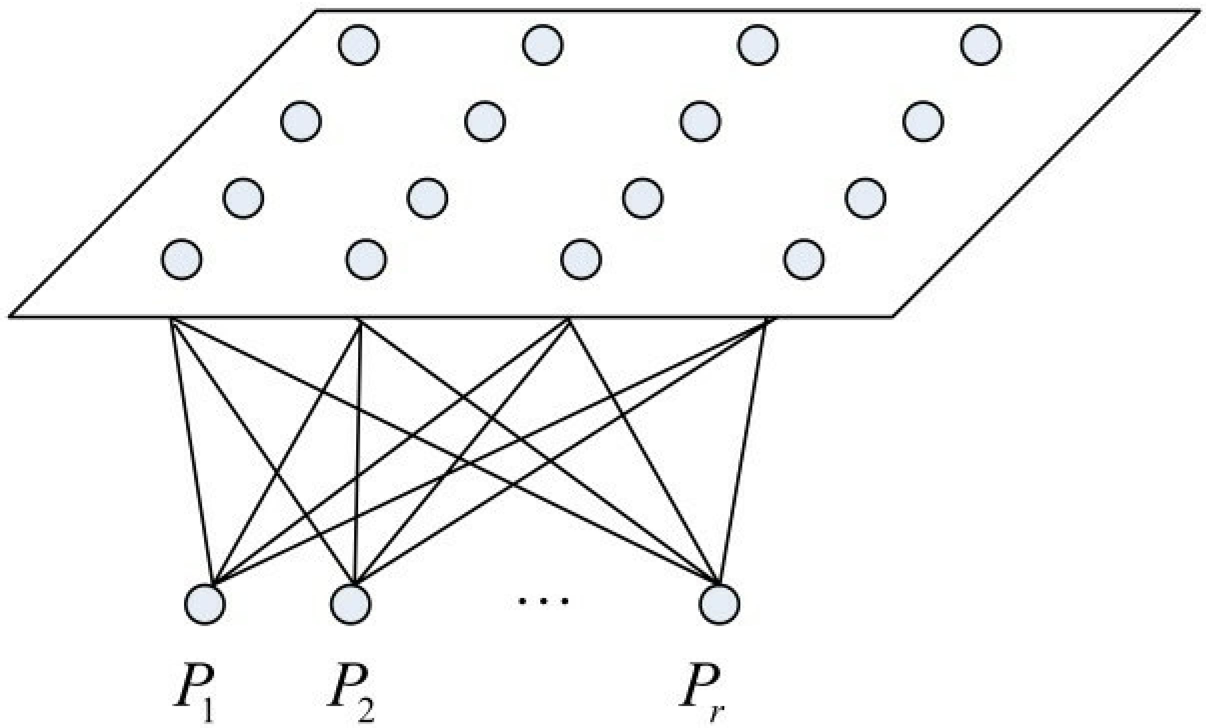


图10-7 二维自组织特征映射神经网络结构图

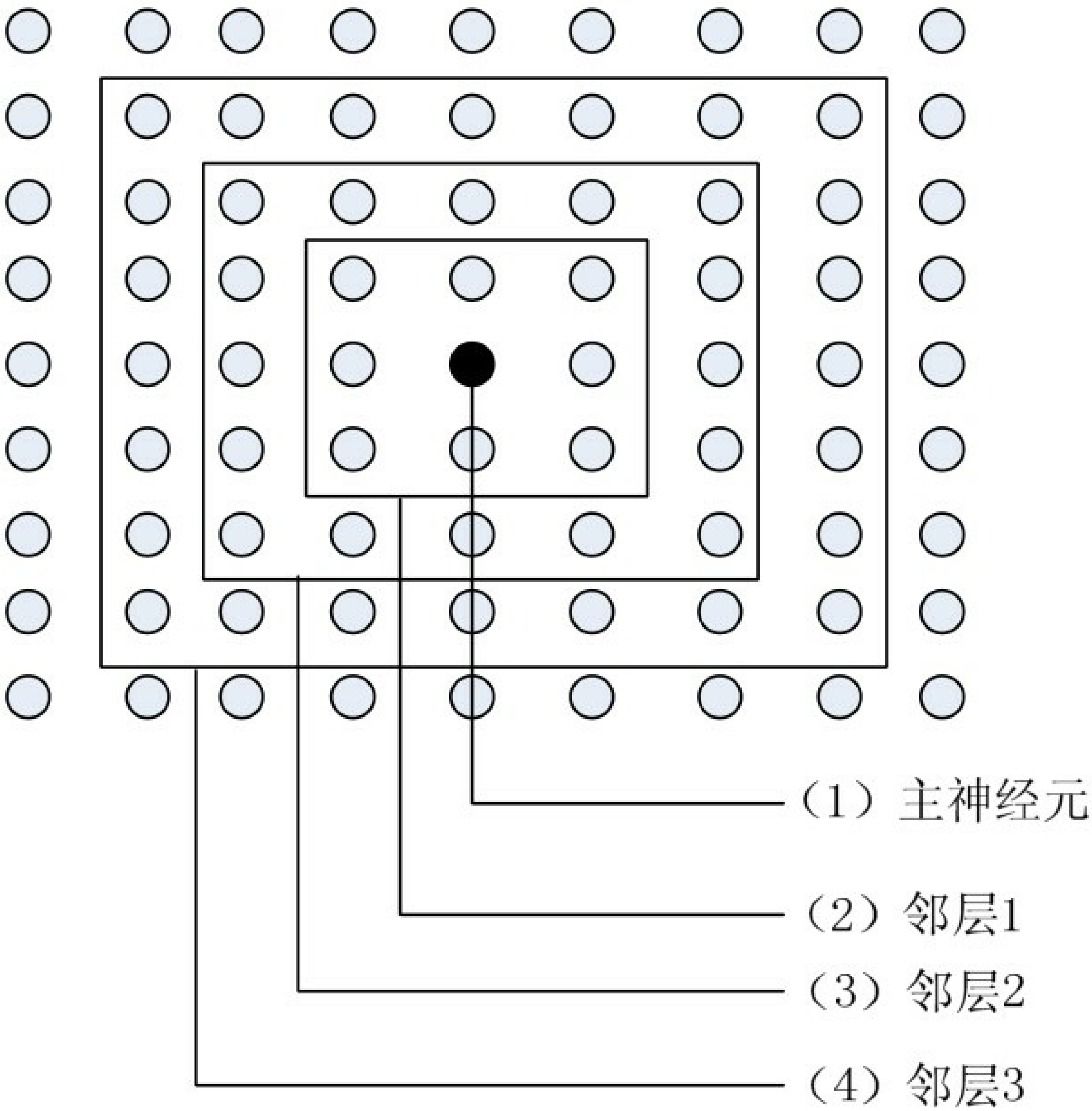


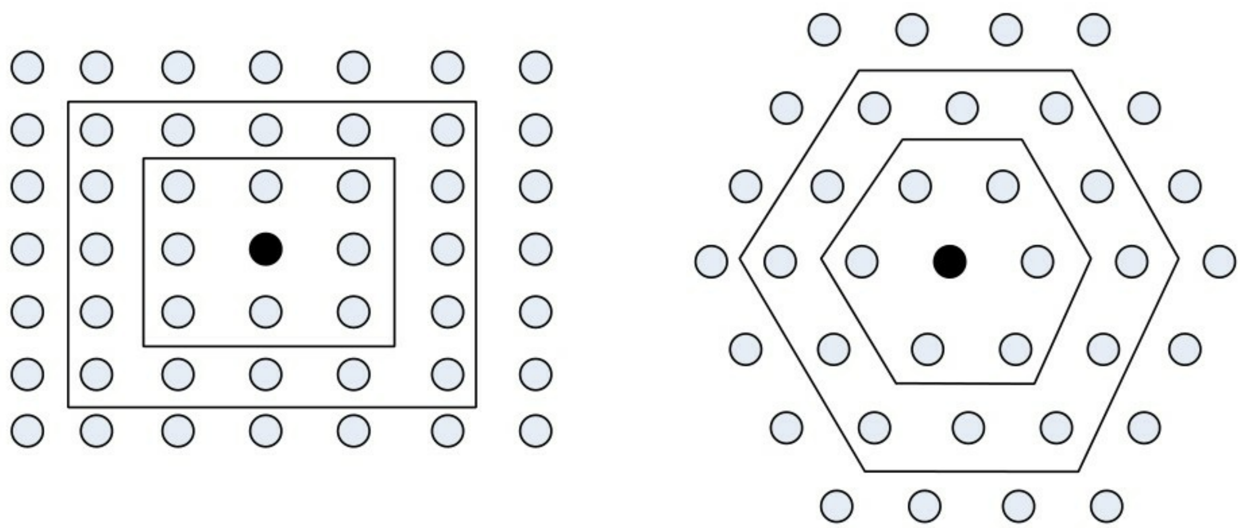
图10-8 二维神经元层示意图

一个直径为1的邻域包括主神经元及它的直接周围神经元所组成的区域；直径为2的邻域包括直径1的神经元以及它们的邻域。图10-8中主神经元的位置是通过从左上端第一列开始顺序从左到右，从上到下找到的。如10×10神经元层，其主神经元位于46。

特性图的激活函数也是二值型函数，同竞争网络一样可以取偏置b为零或固定为一常数。竞争层的竞争结果，不仅使加权输入和为最大值

者获胜而输出为 1，同时也使获胜节点周围的邻域也同时输出为1。

另外，在权值调整的方式上，特性图网络不仅调整与获胜节点相连的权值，而且对获胜节点邻域节点的权值也进行调整，即使其周围 Dk的区域内神经元在不同程度上也得到兴奋，在Dk以外的神经元都被抑制，这个Dk区域可以是以获胜节点为中心的正方形，也可以为六角形，如图10-9所示。



(a) 正方形域　　(b)六边形域

图10-9 二维网络邻域形状

对于一维输出，Dk则为以k为中心的下上邻点。

在MATLAB工具箱中有一个求获胜神经元的邻域的函数，它所需要的参数为：获胜神经元位置i，竞争层的大小和邻域的直径N（如果N_p被省去，意味着直径为1），邻域函数返回一个行矢量，包括N_p直径邻域中的所有神经元，其中包括主神经元本身在内。

在二维竞争层中，邻域函数为neighb2d，竞争层的大小是其宽度x和高度y，层中神经元总数S应当是x与y的乘积。函数neighb2d的用法如下。

Np=[x y]；

in＝neighb2d(i, Np, N)；

对于一维竞争层，其中的邻层函数为neighb1d，确定竞争层大小的参数就是神经元数S。

Np=[S]；

in＝neighb1d(i，Np，N)；

除了正方形以外，我们可以用函数neighb2d.m产生一个具有任意形状的邻层。此时，对新邻域函数的唯一限制是变量列表必须与上述两个函数相同。不过，邻层函数的行矢量可以包含任何常数。

10.2.2 SOM权值调整域

SOM网采用的算法称为Kohonen算法，它是在“胜者为王”（Winner-Take-All，WTA）学习规则基础上加以改进的，主要区别是调整权向量与侧抑制的方式不同。

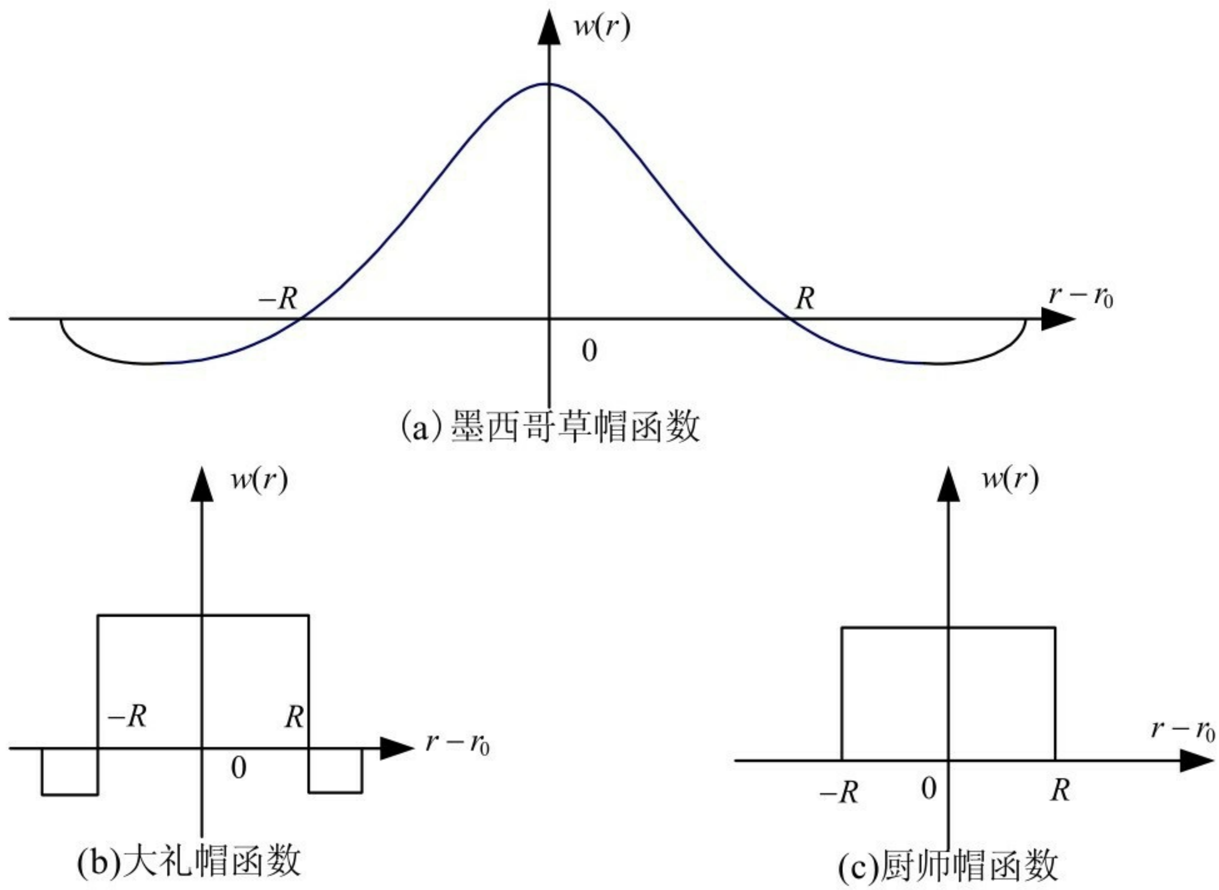
（1）WTA。侧抑制是“封杀”式的。只有获胜神经元可以调整其权值，其他神经元都无权调整。

（2）Kohonen 算法：获胜的神经元对其邻近神经元的影响是由近及远，由兴奋逐渐变为抑制。换句话说，不仅获胜神经元要调整权值，它周围的神经元也要不同程度调整权向量。常见的调整方式有如下几种。

·墨西哥草帽函数。获胜节点有最大的权值调整量，临近的节点有稍小的调整量，离获胜节点距离越大，权值调整量越小，直到某一距离d_0时，权值调整量为零；当距离再远一些时，权值调整量稍负，更远又回到零。如图10-10（a）所示。

·大礼帽函数。它是墨西哥草帽函数的一种简化，如图10-10（b）所示。

·厨师帽函数。它是大礼帽函数的一种简化，如图10-10（c）所示。



(a) 墨西哥草帽函数

(b)大礼帽函数

(c)厨师帽函数

图10-10 权值调整函数

以获胜神经元为中心设定一个邻域半径R，该半径固定的范围称为优胜邻域。在SOM网学习方法中，优胜邻域内的所有神经元，均按其离开获胜神经元距离的远近不同程度调整权值。优胜邻域开始定的较大，但其大小随着训练次数的增加不断收缩，最终收缩到半径为零。

10.2.3 SOM网络运行原理

SOM网络的运行分训练和工作两个阶段。在训练阶段，网络随机输入训练集中的样本，对某个特定的输入模式，输出层会有某个节点产生最大响应而获胜，而在训练开始阶段，输出层哪个位置的节点将对哪类输入模式产生最大响应是不确定的。

当输入模式的类别改变时，二维平面的获胜节点也会改变。获胜节

点周围的节点因侧向相互兴奋作用也产生较大影响，于是获胜节点及其优胜邻域内的所有节点所连接的权向量均向输入方向作不同程度的调整，调整力度依邻域内各节点距离获胜节点的远近而逐渐减小。

网络通过自组织方式，用大量训练样本调整网络权值，最后使输出层各节点成为对特定模式类敏感的神经元，对应的内星权向量成为各输入模式的中心向量。并且当两个模式类的特征接近时，代表这两类的节点在位置上也接近。从而在输出层形成能反应样本模式类分布情况的有序特征图。

10.2.4 网络的训练过程

自组织特征映射神经网络在训练开始时和普通的竞争网络一样，其输入节点竞争的胜利者代表某类模式。然后定义获胜节点的邻域节点，即以获胜节点为中心的某一半径内的所有节点，并对与其相似的权矩阵进行调整。随着训练的继续进行，获胜节点 k 的半径将逐渐变小，直到最后只包含获胜节点k本身。

也就是说，在训练的初始阶段，不但对获胜的节点作权值的调整，而且对其周围较大范围内的几何邻接节点也作相应的调整，而随着训练过程的进行，与获胜输出节点相连的权矩阵就越来越接近其所代表的模式类，此时，需要对获胜节点进行较细致的权矩阵调整。

同时，只对其几何邻接较接近的节点进行相应的调整。这样，在训练结束后，几何上相近的输出节点所连接的权矢量既有联系（即类似性），又相互有区别，保证了对于某一类输入模式、获胜节点能作出最大的响应。而相邻节点作出较少响应。几何上相邻的节点代表特征上相似的模式类别。

特性图的初始权值一般被设置得很小，例如初始化一个特性图用R个输入和S个神经元的过程可为

W=randns(S,R)·0.1

由上可知，特性图不同于常规竞争学习网络那样修正其权值，除了修正获胜权值外，特性图还修正它的邻域权值。结果是邻域的神经元也逐渐趋于相似的权矢量，并对相似的输入矢量作出响应。

网络训练过程如下。

（1）初始化。

① 由输入矢量确定网络结构。

[R, Q]＝size(P)

② 设置网络竞争层神经元节点。

一维S 或二维的宽X 和高Y，S＝X·Y

③ 将输入模式P 作归一化处理。

P＝normc(P)

④ 归一随机化处理初始权值。

W＝rands(S, R)·0.1

设置最大循环次数（此数根据输入数据的数目而乘一个倍数所得）max_cycle；基本学习速率lr一般取0.01～0.3，视具体情况而定；最大邻层数max_neighb分为一维max_neighb＝S−1；二维max_neighb＝max([X Y])−1。

（2）循环训练。

for cycle=1:max_cycle

① 学习速率线性下降。

x＝cycle/max_cycle;

LR＝(1—x)*lr;

这使得学习速率随着循环次数的增加，从lr·(max_cycle-1)/max_cycle逐渐降至0。

② 邻层数递减。

n=max([ceil(max_neighb*(1-x*4)) 1]);

③ 计算输入矢量加权和，并通过竞争求出获胜节点。

A＝compet(W*P)；

④ 根据获胜节点求出相邻层（以二维为例），并进行权值修正。

i=find(A＝＝1)；

in=neighb2d(i, [X, Y], n);

dW＝learn(W, P, in, LR)

W＝W+dw；

（3）输出或显示结果。

在训练过程中，神经元参与彼此的竞争活动，而具有最大输出的神经元节点是获胜者。

该获胜节点具有抑制其他竞争者和激活其近邻节点的能力，但是只有获胜节点才允许有输出，也只有获胜节点和其邻近节点的权值允许被调节。获胜节点的近邻节点的范围在训练过程中是可变的。

在训练开始时，一般将邻近范围取得较大，随着训练的进行，其邻近范围逐渐缩小。因为只有获胜节点是输入图形的最佳匹配，所以说自组织特征映射神经网络模仿了输入图形的分布，或者说该网络能提取输入图形的特征，把输入图形特征相似地分到一类，由某一获胜节点表示。

MATLAB工具箱中用于训练设计自组织特征映射神经网络权矢量的函数为trainfm.m，它能执行上述的训练步骤的全过程，仅需要给它输入4个训练参数：显示频率disp_freq，最大训练周期max_cycle，学习速率lr以及最大邻域数max_neighb。

TF=[disp_freq max_cycle lr max_neighb]

W=trainfm(W,P,TF)

最大邻域数应设置为层的最大直径数减去l。例如，对一维特性图max_neighb应为S−1；对二维特性图，其层神经元宽为X高为Y时，max_neighb应当等于两者中的较大值减去1。不论哪个神经元被选为主

神经元，最大邻域数总是包括了层中的所有神经元。

函数trainfm.m的训练开始于学习速度lr和最大邻域max_neighb，然后，其学习速率线性地递减，以致于最后的训练周期里的学习速率为0。邻域数也是线性地减少，但在达到四分之一训练周期后，其值保持为最大值1直到最终训练结束。

这种安排允许输出神经元在初始阶段的活动范围较大，并使其全体都趋于输入矢量出现的区域。然后，随着邻层的减少至 1，图形趋于所出现的输入矢量的自身。一旦邻域大小为 1，图形则已被很好地排了序。学习速率也是在一个较长的时间里进行衰减，以给神经元足够的时间展开。

如果输入矢量是以遍历整个输入空间的概率出现，特性图训练的最终结果将是以接近于输入矢量之间等距离的位置排列；如果输入矢量是以遍历输入空间的变化频率出现，特性图将趋于将神经元定位为一个正比于输入矢量频率的面积。由此，特性图通过被训练，能够学习输入矢量的类型与其贡献的大小，而将其分类。

给定一个特性图的权矩阵 W，它的邻域函数 F（作为一个字符串），以及竞争层大小N_p，可以用函数plotmap画出特性图。

10.3 自适应共振理论神经网络

自适应共振理论（Adaptive Resonance Theory，简称 ART）神经网络既能模拟人脑的可塑性，可以学习知识，又能模拟人脑的稳定性，学习新的知识但不破坏原有知识，即这种网络不仅能记忆新的知识，而且还保留已记忆的内容，这与某些神经网络是不同的。

10.3.1 自适应共振理论神经网络概述

这种网络主要依据 ART，用生物神经细胞自兴奋与侧抑制的动力学原理指导学习，输出层各神经元竞争对输入模式的响应。

在竞争中可能采用侧抑制的方法，最后只有一个神经元获胜，获胜神经元则代表该输入模式的类别，权值的调整只在与获胜神经元关联的连接权值中进行，通过网络双向连接权值的记忆和比较，来完成对输入模式的记忆、回想，并以同样的方式实现模式的识别。

当提供给网络的输入模式是一个网络已记忆的或与已记忆的模式十分相似的模式时，网络会把这个模式回想出来，并提供正确的分类；如果输入模式是网络不曾记忆的新模式，则网络将在不影响原有记忆的前提下，将这个模式记下来，并分配一个尚未使用过的输出层神经元作为这一记忆模式的记忆分类标志。

10.3.2 ART网络的结构及特点

ART网络主要有ARTl和ART2两种模型，其主要区别是前者为二值输入，后者为模拟输入。这里主要介绍ARTl模型结构。

ARTl网络结构如图10-11所示。

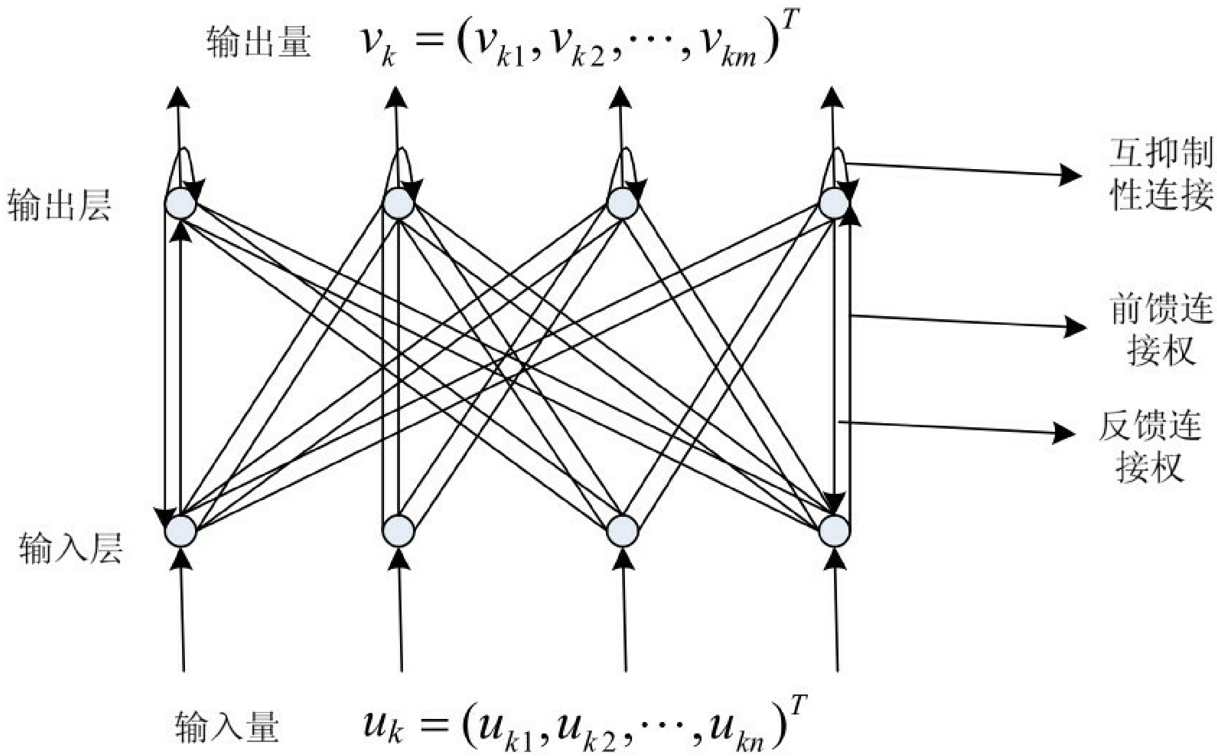


图10-11 ARTl网络结构

网络分为输入层和输出层。输入层有n个神经元，输出层有m个神经元，输入层和输出层之间为双向连接。

ARTl 网络的学习和工作是通过反复地将输入学习模式由输入层向输出层自下而上地短时记忆和由输出层向输入层自上而下地长期记忆和比较来实现的。

当这种记忆和比较达到共振时，输出矢量可以正确地反映出输入学习模式的分类，且网络原有的记忆不受影响。至此，对一个输入模式的记忆和分类即告完成。

ART具有很多特点，主要分为以下几点。

·可以完成实时学习，并适应非平稳环境。

·对已学习过的对象具有稳定快速的识别能力，同时又能迅速地记忆未学习的新对象。

·具有自归一能力，根据某些特征在全体中所占的比例，有时将其

作为关键特征，有时又作为噪声处理。

·可实现非监督学习。

·容量不受输入模式数的限制，也不要求存储对象是正交的。

·分类具有脆弱性，一旦输出层某神经元失效，将导致该神经元所代表的类别信息消失。

10.4 学习向量量化神经网络

学习向量量化LVQ（Learning Vector Quantization）神经网络由芬兰学者Teuvo　Kohonen提出，属于前向有监督神经网络类型，在模式识别和优化领域有着广泛的应用。

10.4.1 LVQ神经网络结构

LVQ神经网络由输入层、隐含层和输出层3层组成，输入层与隐含层间为完全连接，每个输出层神经元与隐含层神经元的不同组相连接。隐含层和输出层神经元之间的连接权值固定为1。在网络训练过程中，输入层和隐含层神经元间的权值被修改。

LVQ神经网络的结构图如图10-12所示。

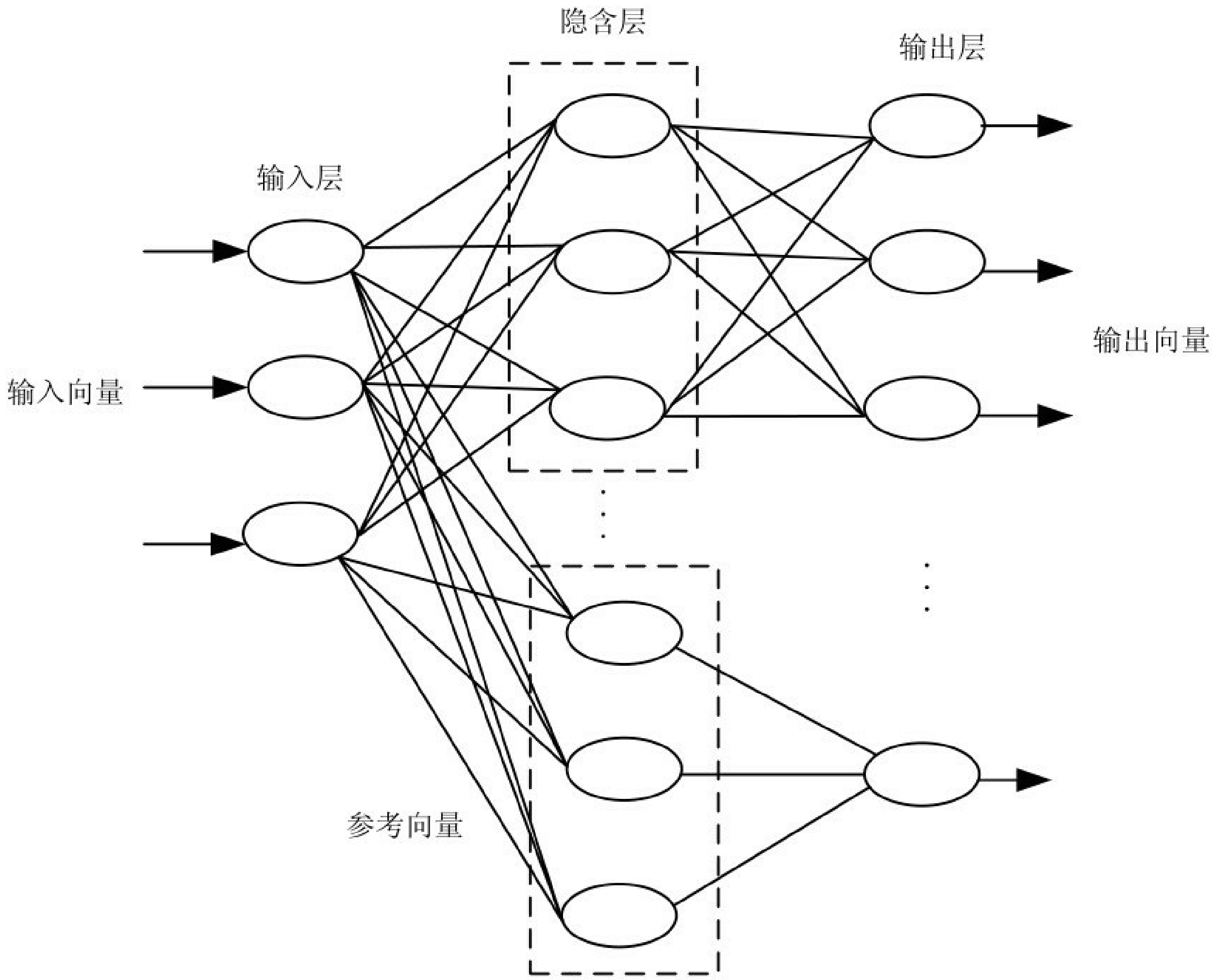


图10-12 LVQ神经网络结构图

当某个输入模式被输入网络时，最接近输入模式的隐含神经元因获得激发而赢得竞争，因而允许它产生一个“1”，而其他隐含层神经元都被迫产生“0”。与包含获胜神经元的隐含层神经元组相连接的输出神经元也发出“1”，而其他输出神经元均发出“0”。

10.4.2 LVQ神经网络算法

LVQ算法具体步骤如下。

（1）网络初始化。

用较小的随机数设定输入层和隐含层之间的权值初始值。

（2）输入向量的输入。

将输入向量 $X=\left[x_1, x_2, x_3, \cdots, x_n\right]^{\mathrm{T}}$ 送入到输入层。

（3）计算隐含层权值向量与输入向量的距离。

隐含层神经元和输入向量的距离，与自组织化映射的情况相同，由下式给出。

$$d_j=\sqrt{\sum_{i=1}^{n}(x_i-w_{ij})^2}$$

（4）选择与权值向量的距离最小的神经元。

计算并选择输入向量和权值向量的距离最小的神经元，并把其称为胜出神经元，记为j*。

（5）更新连接权值。

如果胜出神经元和预先指定的分类一致，称为正确分类，否则称为不正确分类。正确分类和不正确分类时权值的调整量分别使用以下公式。

$$\Delta w_{ij}=\begin{cases}+\eta(x_i-w_{ij})\\-\eta(x_i-w_{ij})\end{cases}$$

（6）判断是否满足预先设定的最大迭代次数，满足时算法结束，否则返回 2，进入下一轮学习。

10.5 竞争型神经网络工具箱函数

竞争型神经网络是神经网络领域中最吸引人的话题之一。这种结构的网络能够从输入信息中找出规律以及关系，并且根据这些规律来相应地调整均衡网络，使得以后的输出与之相适应。

自组织竞争型神经网络能够识别成组的相似向量，常用于进行模式分类。自组织特征映射神经网络不但能够像自组织竞争神经网络一样学习输入的分布情况，而且可以学习进行训练神经网络的拓扑结构。

MATLAB R2013a 神经网络工具箱为竞争型神经网络提供了大量的函数工具。本节将详细介绍这些函数的功能、调用格式和注意事项等。

竞争型神经网络常用的函数见表10-2。

表10-2 竞争型神经网络常用的函数

函数名称	函数功能
compet( )	竞争传输函数
nngenc( )	产生一定类别的样本向量
nbgrid( )	用栅格距离表示的领域矩阵
dist( )	欧氏距离权值函数
nbman( )	用 Manhattan 距离表示的领域矩阵
plotsm( )	绘制竞争网络的权值矢量
initc( )	初始化竞争神经网络
trainc( )	训练竞争神经网络
simuc( )	仿真竞争神经网络
newc( )	建立一个竞争神经网络
initsm( )	初始化自组织特征映射网络
learnk ( )	Kohonen 权值学习规则函数
learnis ( )	Instar 权值学习规则函数
learnos ( )	Outstar 权值学习规则函数
learnh ( )	Hebb 权值学习规则函数
learnhd( )	衰减的 Hebb 权值学习规则函数
learnsom( )	自组织特征映射权值学习函数
plotsom( )	绘制自组织特征映射网络的权值矢量
trainsm( )	利用 Kohonen 规则训练自组织特征映射网络
simusm( )	仿真自组织特征映射网络
newsom( )	创建一个自组织特征映射神经网络
negdist( )	对输入矢量进行加权计算
netsum( )	计算网络输入矢量和
midpoint ( )	中点权值初始化函数
mandist( )	Manhattan 距离权值函数
initlvq( )	LVQ 神经网络的初始化函数
plotvec( )	用不同的颜色画矢量函数

下面介绍表10-2中几个重要的函数。

（1）竞争传输函数compet( )。

该函数将神经网络输入进行转换，使网络输入最大的神经元输出为1，而其余的神经元输出为0。函数compet ( )调用格式如下。

Y=compet(X) 或 Y=compet(Z,b)

下面代码是函数compet的使用示例。

```
n = [0; 1;-0.5; 0.5];
a = compet(n);
subplot(2,1,1), bar(n), ylabel('n')
subplot(2,1,2), bar(a), ylabel('a')
```

在MATLAB的Command Window窗口，输入以上代码，得到结果如图10-13 所示。

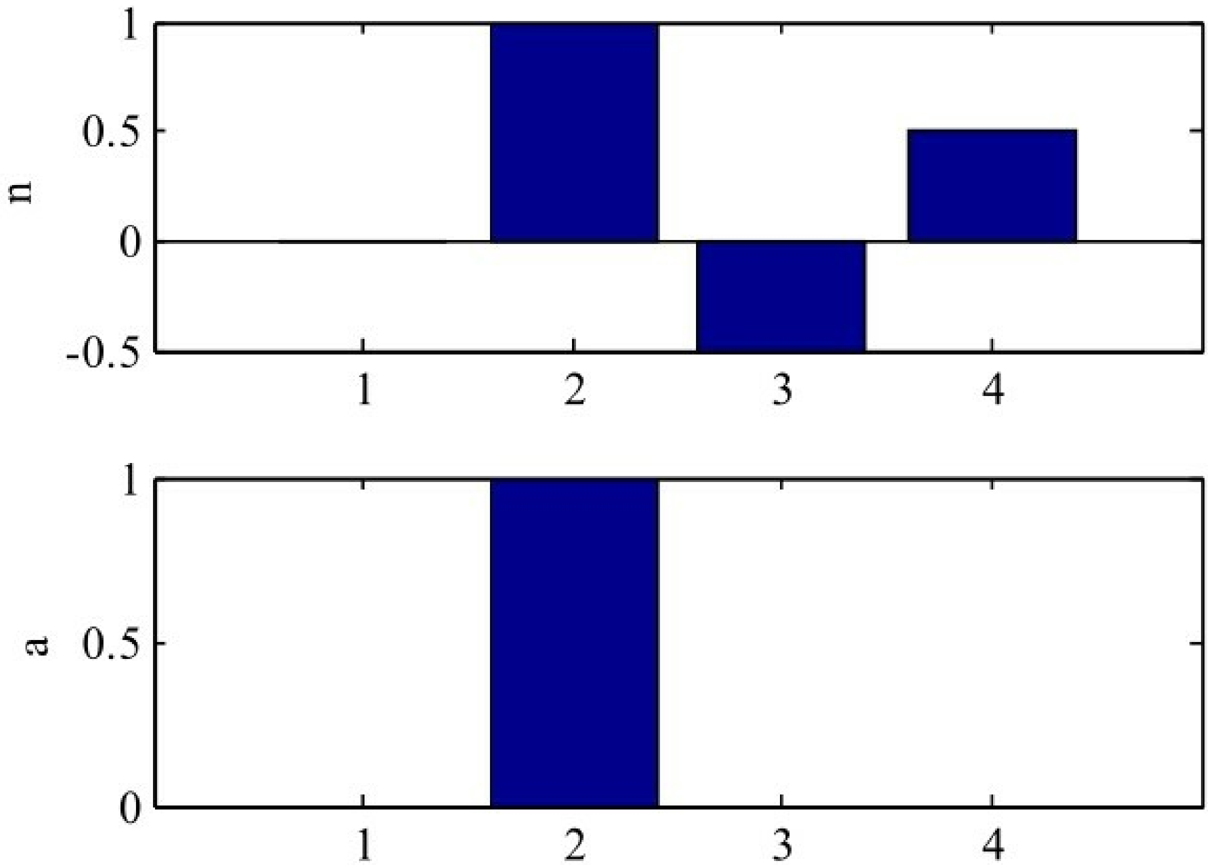


图10-13 函数compet使用示例结果

从图10-13中可以看出，输入向量n的第2个数最大。使用函数compet后得到的输出变量中，第2个数为1，其他为0。

如果使用full命令，可以更明显地得到输出变量的值如下所示。

```
ans =
   0
```

1

0

0

（2）向量产生函数nngenc( )。

该函数可以产生一定类别的样本，其调用格式如下。

X=nngenc(C,clusters,points,std_dev)

【例10-4】利用nngenc函数产生一定类别的样本，再建立一个竞争型神经网络来区分这些类。

解：首先使用函数compet产生一定类别的样本，具体代码如下所示。

```
bounds= [01; 01];           %设定边界
clusters= 8;                % 设定簇的数量.
points=10;                  % 设定每个簇的点数.
std_dev= 0.05;              % 每个簇的标准偏差.
x= nngenc(bounds,clusters,points,std_dev); %创建输入向量X.
% 绘制输入向量X.
plot(x(1,:),x(2,:),'+r');
title('Input Vectors');
xlabel('x(1)');
ylabel('x(2)');
```

输入以上代码，得到结果如图10-14所示。

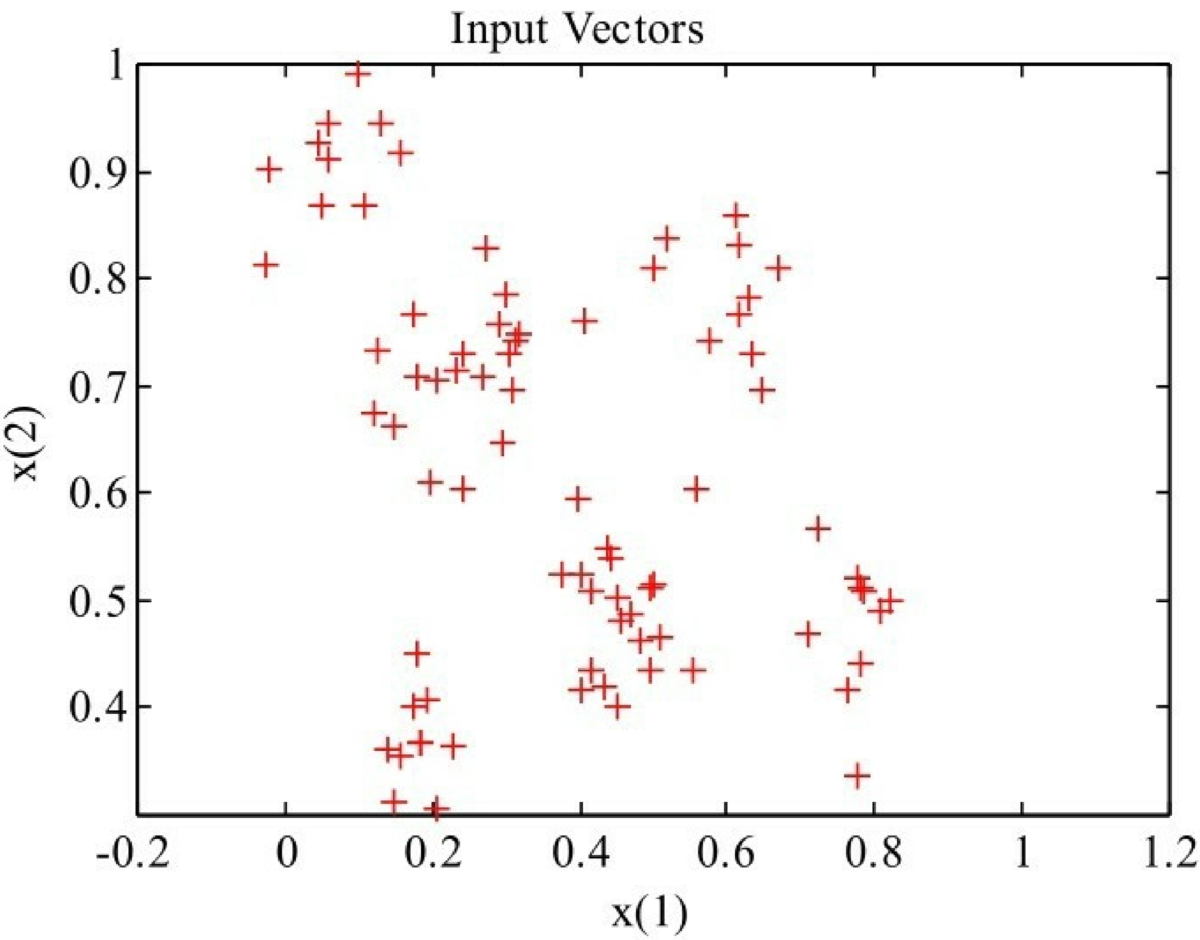


图10-14 使用函数compet产生一定类别的输入样本

建立一个竞争型神经网络，并画出一个分类点，使其出现在输入向量集群中。MATLAB代码如下所示。

```
net = competlayer(8,.1);
net = configure(net,x);
w = net.IW{1};
plot(x(1,:),x(2,:),'+r');
hold on;
circles = plot(w(:,1),w(:,2),'ob');
```

运行以上代码，得到结果如图10-15所示。

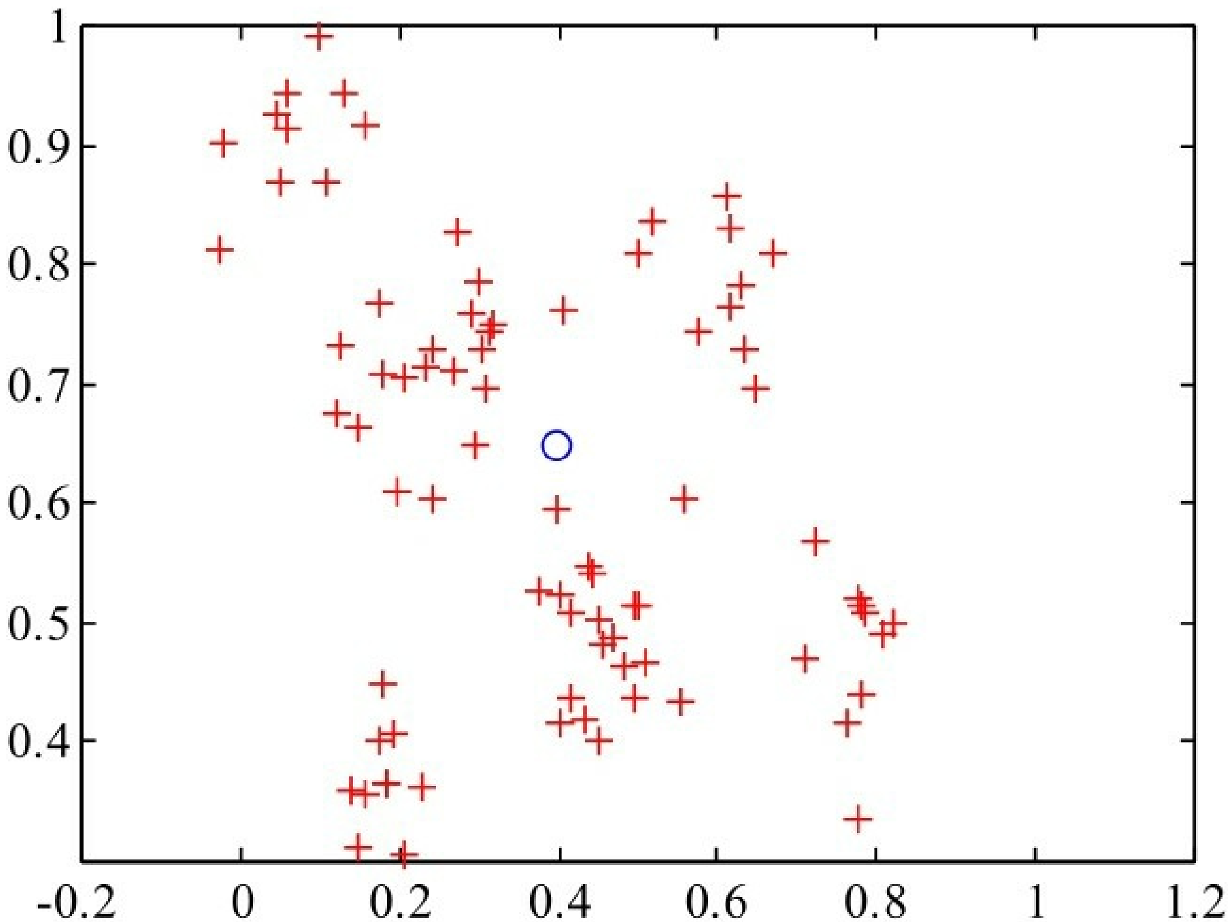


图10-15 竞争型神经网络得到的输入向量中的分类点

训练竞争型神经网络，并更新网络权重，其MATLAB代码如下所示。

```
net.trainParam.epochs = 7;
net = train(net,x);
w = net.IW{1};
delete(circles);
plot(w(:,1),w(:,2),'ob');
```

得到分类点的结果如图10-16所示。从图中可以看到，随机产生的向量分成了8类。

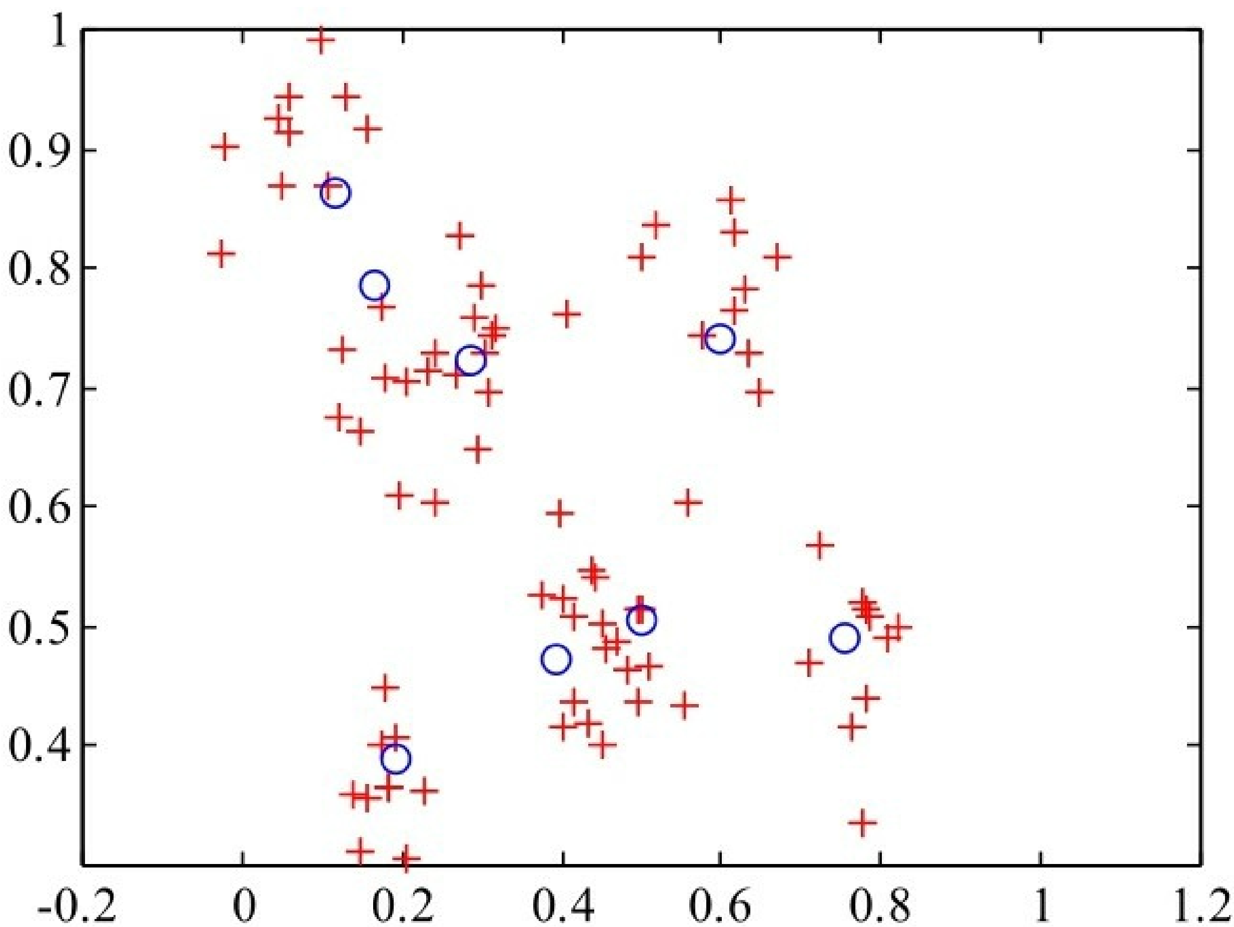


图10-16 训练竞争型神经网络得到的结果

现在可以使用竞争层作为一个分类器，其中每个神经元对应于一个不同的类别。这里定义输入向量x1= [0;0.2]，运行以下代码。

```
x1 = [0; 0.2];
y = net(x1
```

得到表明神经元输出的结果如下。

```
y =
   1
   0
   0
   0
   0
```

0

0

0

（3）训练竞争层函数trainc( )。

函数trainc( ) 训练竞争层，对一组输入矢量分类，它从一组输入矢量中随机选取一个矢量，然后找出输入最大的神经元，并按Kohonen准则修改权值，从而训练竞争层网络。函数调用格式如下。

[W,b]=trainc(w,X,tp)

（4）Konohen 权值学习规则函数learnk( )。

函数learnk( )根据Konohen 相关准则计算网络层的权值变化矩阵，其学习通过调整神经元的权值等于当前输入。使神经元存储输入，用于以后的识别，即Δw(i,j)=η(x(j)-w(i,j))。函数调用格式如下。

[dW,NLS]=learnk(W,X,Z,N,A,T,E,gW,gA,D,LP,LS)

其中，W 为S·R 权值矩阵。

X 为R·Q输入向量。

Z 为S·Q加权输入向量。

N 为S·Q神经网络输入向量。

A 为S·Q输出向量。

T 为S·Q层的目标向量。

E 为S·Q层的误差向量。

gW 为S·R相对于网络性能的加权梯度向量。

gA 为S·Q相对于网络性能的输出梯度向量。

D为S·S 神经元距离矩阵。

LP为学习参数（或者学习速率）。

LS为学习状态。

假如定义一个神经网络输入变量、输出变量、权值和学习速率，代码如下所示。

```
p = rand(2,1);
a = rand(3,1);
w = rand(3,2);
lp.lr = 0.5;
```

使用学习函数learnk。

```
dW = learnk(w,p,[],[],a,[],[],[],[],[],lp,[])
```

得到其权值调整量如下。

```
dW =
   0.1947    -0.2191
   -0.0485    -0.0583
   -0.1208    -0.3241
```

（5）Instar权值学习规则函数learnis( )。

函数learnis( )根据Instar相关准则计算网络层的权值变化矩阵，其学习用一个正比于神经网络的学习速率来调整权值，学习一个新的矢量使之等于当前输入。这样任何使 Instar层引起高输出的变化，都会导致网络根据当前的输入向量学习这种变化。最终相同的输入使网络有明显不同的输出，即$\Delta w(i,j)=\eta y(i)(x(j)-w(i,j))$。函数调用格式如下。

```
[dW,NLS]=learnis(W,X,Z,N,A,T,E,gW,gA,D,LP,LS)
```

该函数的用法和各个参数的定义同函数learnk( )。

假如定义一个神经网络输入变量、输出变量、权值和学习速率，代码如下所示。

```
p = rand(3,1);
a = rand(4,1);
w = rand(4,3);
lp.lr = 0.6;
```

使用学习函数learnis。

```
dW = learnis(w,p,[],[],a,[],[],[],[],[],lp,[])
```

得到其权值调整量如下。

```
dW =
   -0.0071   -0.0426   0.1159
   -0.0504   0.0047   0.0531
   0.0290   -0.0655   0.4688
   -0.0157   -0.0108   0.0181
```

（6）Outstar权值学习规则函数learnos( )。

函数 learnos( )根据 Outstar相关准则计算网络层的权值变化矩阵，Outstar网络层的权可以看作是与网络层的输入矢量一样多的长期存储器。通常，Outstar层是线性的，允许输入权值按线性层学习输入矢量。因此，存储在输入权值中的矢量可通过激活该输入而得到，即 $\Delta w(i,j)=\eta(y(i)-w(i,j))/x(j)$。函数调用格式如下。

```
[dW,NLS]=learnos(W,X,Z,N,A,T,E,gW,gA,D,LP,LS)
```

该函数的用法和各个参数的定义同函数learnk( )。

假如定义一个神经网络输入变量、输出变量、权值和学习速率，代码如下所示。

```
p = rand(2,1);
a = rand(4,1);
w = rand(4,2);
lp.lr = 0.4;
```

使用学习函数learnos。

```
dW = learnos(w,p,[],[],a,[],[],[],[],[],lp,[])
```

得到其权值调整量如下。

```
dW =
   -0.0039   0.0740
   -0.2334   -0.0438
   0.1526   0.0854
```

0.2702   0.2231

（7）Hebb 权值学习规则函数learnh( )。

Hebb在1943年首次提出了神经元学习规则，他认为两个神经元之间的连接权值的强度与所连接的两个神经元的活化水平成正比。

也就是说，如果一个神经元的输入值大，那么其输出值也大，而且输入和神经元之间的权值也相应增大。其原理可表示为$\Delta w(i,j)=\eta\cdot y(i)\cdot x(j)$。由此可看出，第 j 个输入和第 i个神经元之间的权值的变化量同输入x(j)和输出y(i)的乘积成正比。函数调用格式如下。

```
[dW,NLS]=learnh(W,X,Z,N,A,T,E,gW,gA,D,LP,LS)
```

该函数的用法和各个参数的定义同函数learnk( )。

假如定义一个神经网络输入变量、输出变量、权值和学习速率，代码如下所示。

```
p = rand(2,1);
a = rand(3,1);
lp.lr = 0.5;
```

使用学习函数learnh。

```
dW = learnh([],p,[],[],a,[],[],[],[],[],lp,[])
```

得到其权值调整量如下。

```
dW =
   0.2518    0.1322
   0.2412    0.1266
   0.1045    0.0549
```

（8）衰减的Hebb 权值学习规则函数learnhd( )。

原始的Hebb学习规则对权值矩阵的取值未做任何限制，因而学习后权值可取任意值。为了克服这一弊病，在 Hebb 学习规则的基础上增加一个衰减项，即$\Delta w(i,j)=\eta\cdot y(i)\cdot x(j)-dr\cdot w(i,j)$。

衰减项的加入能够增加网络学习的“记忆”功能，并且能有效地对权

值加以限制，衰减系数dr的取值应该在[0,1]之间。

当dr取为0时，就变成原始的Hebb学习规则，网络学习不具备“记忆”功能；当dr取为1时，网络学习结束后权值取值很小，不过网络能“记忆”前几个循环中学习的内容。

这种改进算法可利用衰减的Hebb 权值学习规则函数learnhd( )来实现。

函数调用格式如下。

```
[dW,NLS]=learnhd(W,X,Z,N,A,T,E,gW,gA,D,LP,LS)
```

该函数的用法和各个参数的定义同函数learnk( )。

假如定义一个神经网络输入变量、输出变量、权值和学习速率，代码如下所示。

```
p = rand(2,1);
a = rand(3,1);
w = rand(3,2);
lp.dr = 0.05;
lp.lr = 0.5;
```

使用学习函数learnhd。

```
dW = learnhd(w,p,[],[],a,[],[],[],[],[],lp,[])
```

得到其权值调整量如下。

```
dW =
    0.0469    0.0096
    0.3581    0.3654
    0.2303    0.2785
```

（9）自组织特征映射权值学习函数learnsom( )。

函数learnsom( )是根据所给出的学习参数LP开始的，其正常状态学习速率LP.order_lr默认值为0.9，正常状态学习步数LP.order_steps默认值为1000，调整状态学习速率LP.tune_lr默认值为0.02，调整状态邻域距离

LP.tune_nd默认值为1。

在网络处于正常状态和调整状态时，学习速率和邻域尺寸都得到更新。函数调用格式如下。

[dW,NLS]=learnsom(W,X,Z,N,A,T,E,gW,gA,D,LP,LS)

该函数的用法和其余参数的定义同函数learnk( )。

函数learnsom( )实例如下所示。

```
p = rand(2,1);
a = rand(6,1);
w = rand(6,2);
pos = hextop(2,3);
d = linkdist(pos);
lp.order_lr = 0.9;
lp.order_steps = 1000;
lp.tune_lr = 0.02;
lp.tune_nd = 1;
ls = [];
[dW,ls] = learnsom(w,p,[],[],a,[],[],[],[],d,lp,ls)
```

在MATLAB中输入以上代码，得到如下所示结果。

```
dW =
   0.0202    0.3628
   0.6174    0.4197
   1.5562    0.9404
   0.2175    0.5402
   0.0737    1.5821
   0.5712    0.4915
ls =
   step: 1
```

nd_max: 2

（10）绘制自组织特征映射网络的权值矢量函数plotsom( )。

函数 plotsom(W,m)用于绘制自组织映射网络的权值图，在每个神经元的权矢量（行）相应的坐标处画一点，表示相邻神经元权值的点，根据邻阵 m 用实线连接起来。即如果M(i,j)≤1，则将神经元i和j用线连接起来。

调用格式如下。

plotsom(W,m)

式中，W为权值矩阵；m为网络邻域。

【例10-5】建立一个输入向量分布在一个二维空间，其变化范围均为[-1 1]，网络结构为4×4 (16 个神经元)的二维自组织特征映射神经网络。

解：建立网络及训练的代码如下所示。

```
clear all
clc
%设定输入
P=rands(10,4);
%建立SOM神经网络
net = newsom(P,[4 4]);
%训练网络
net = train(net,P);
%绘制网络权值变化
figure(1)
plot(P(1,:)',P(2,:)','.g','markersize',20)
hold on
figure(2)
plot(P(1,:)',P(2,:)','.g','markersize',20)
```

```
hold on
plotsom(net.iw{1,1},net.layers{1}.distances)
hold off
```

运行以上代码，得到所建立的神经网络训练窗口如图10-17所示。

单击神经网络训练窗口中的“SOM Weight Positions”按钮，得到图10-18所示的网络权值所在的位置。

权值向量的变化曲线如图10-19所示。

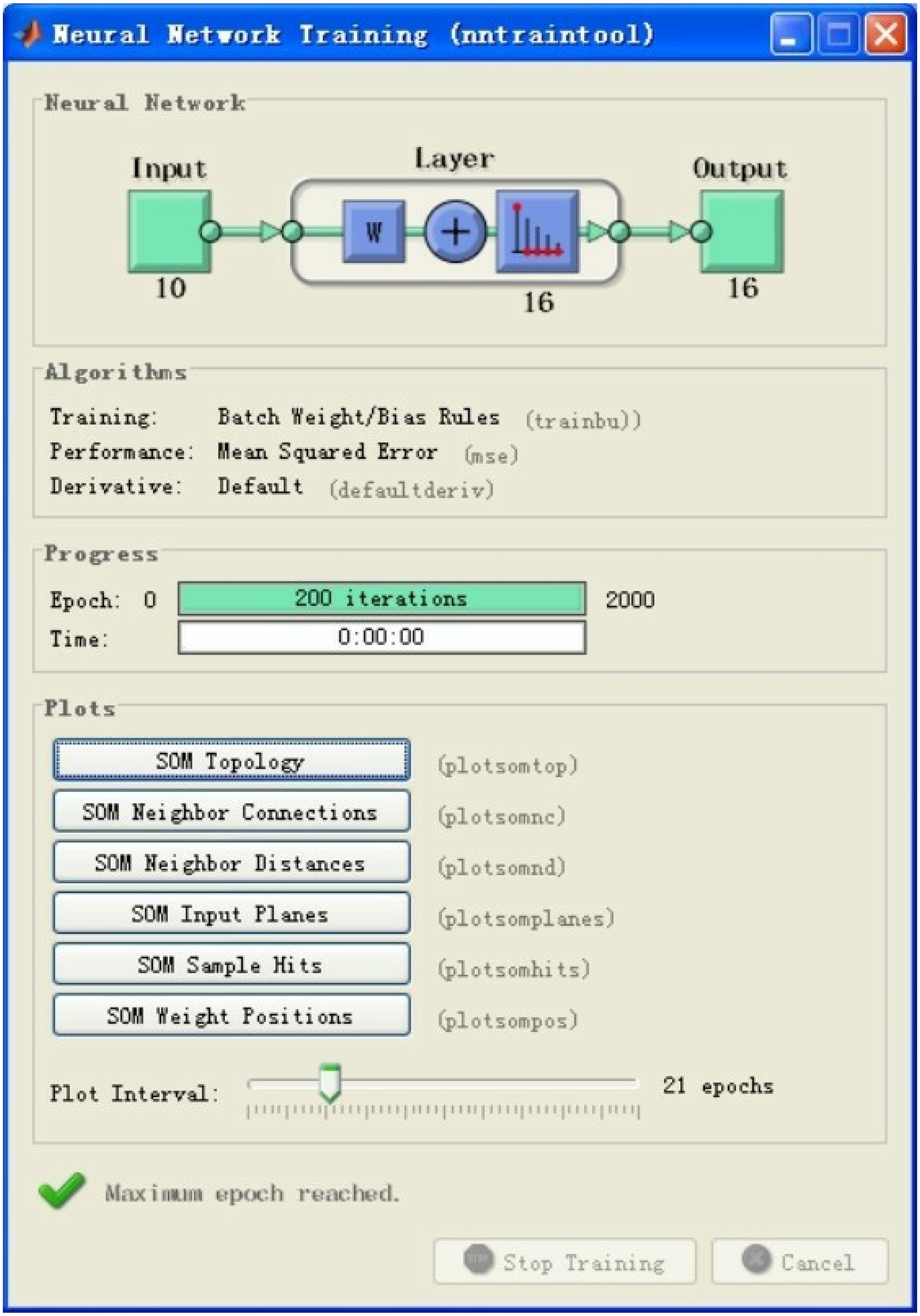


图10-17 神经网络训练窗口

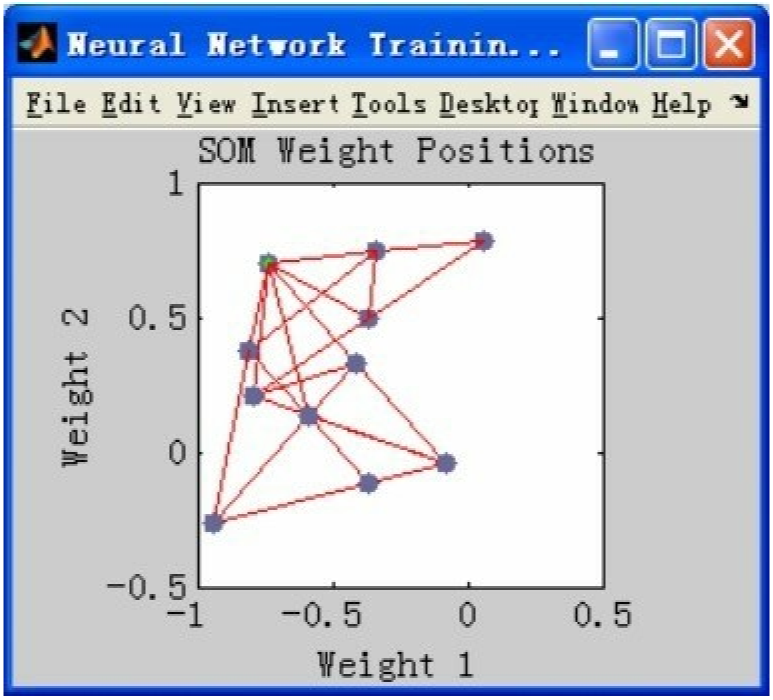


图10-18 训练过程中网络权值所在的位置

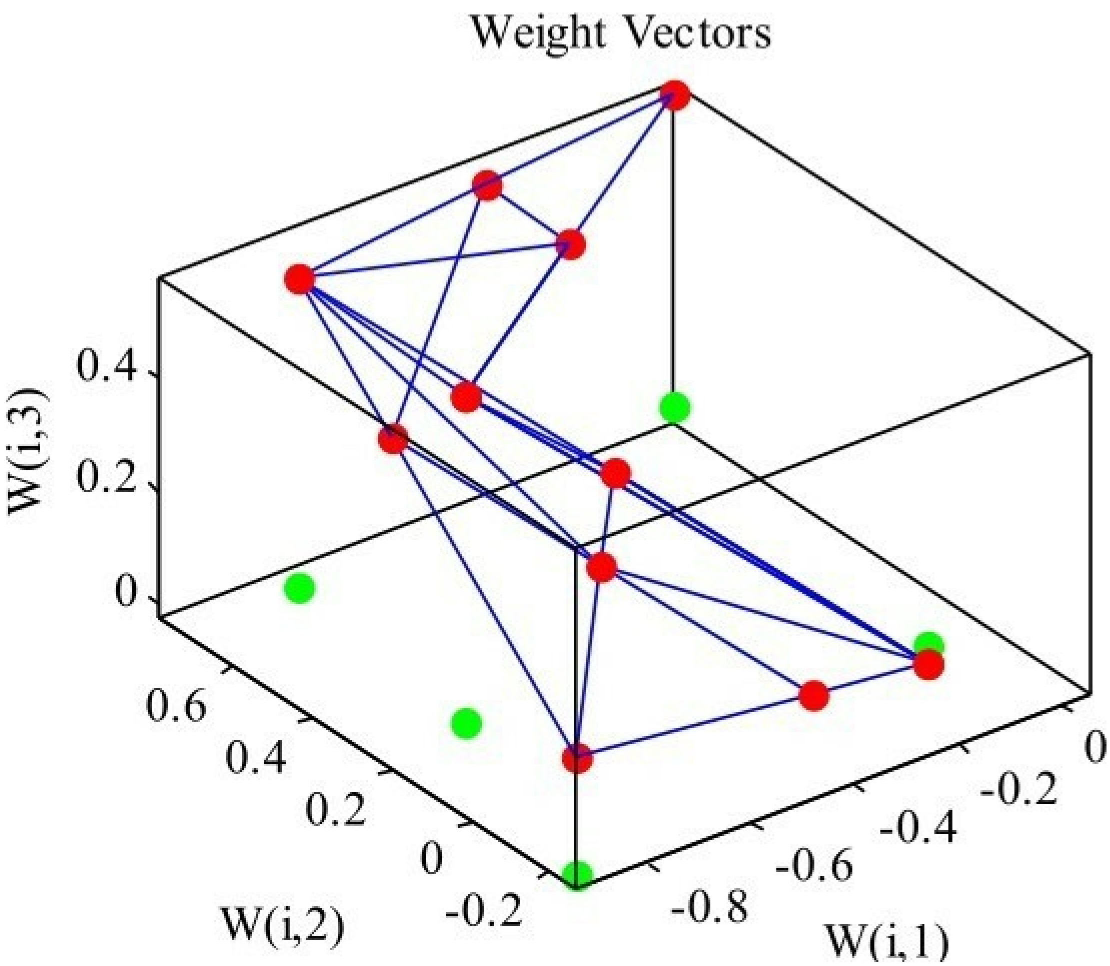


图10-19 权值向量的变化曲线

（11）创建一个自组织特征映射网络函数newsom( )。

利用newsom( ) 函数可建立一个自组织特征映网络。其调用格式如下。

net=newsom (Xr,[d1,d2,…,di])

【例10-6】建立一个输入向量分布在一个二维空间，其变化范围分别为[0 1]和[0 3]，网络结构为4×3 (12 个神经元)的二维自组织特征映射神经网络。

解：建立网络可利用命令newsom，具体代码如下所示。

```
net=newsom([0 1;0 3],[4 3]);
plotsom(net.layers{1}.positions)
```

执行结果如图10-20所示。

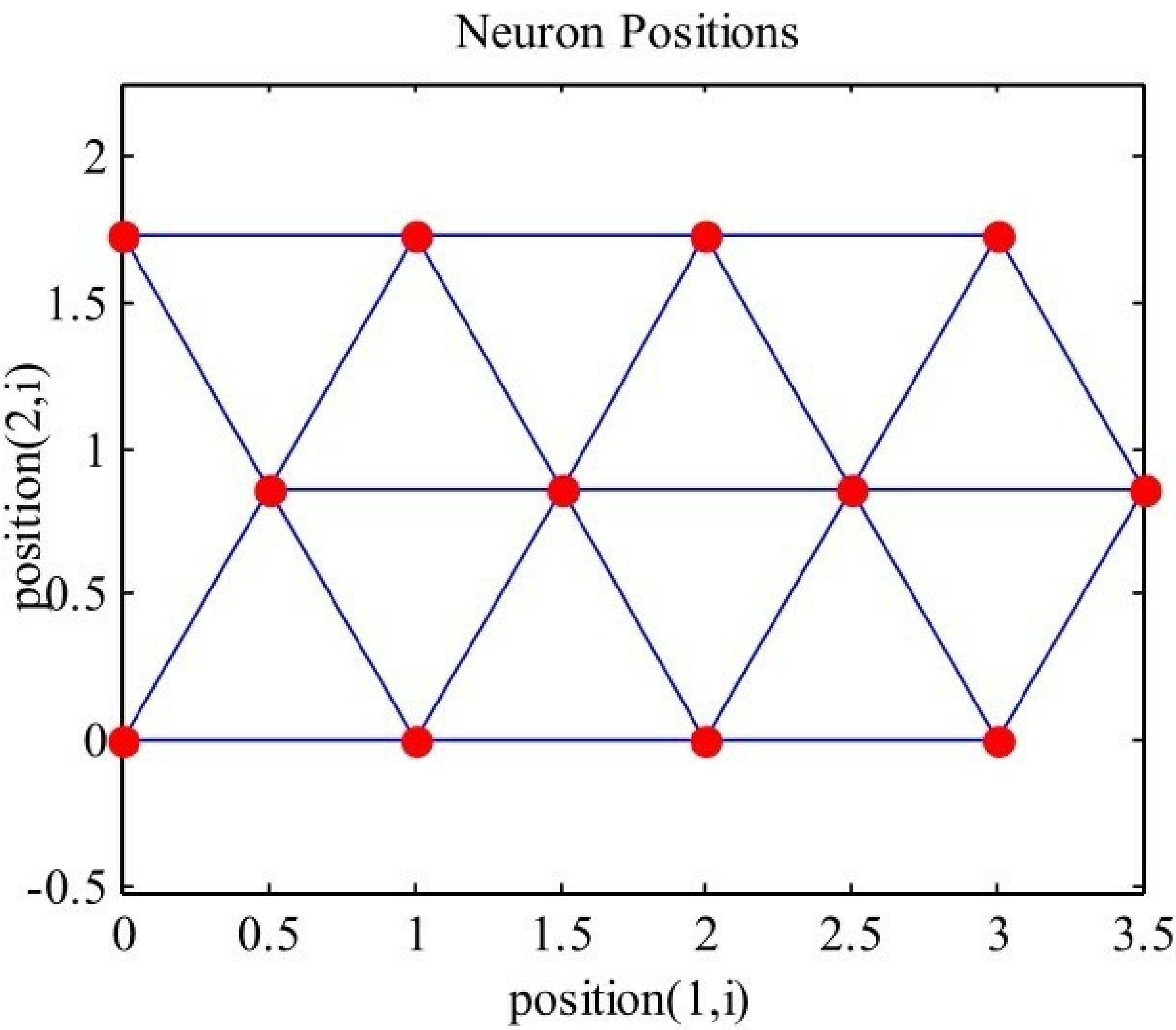


图10-20 神经网络层的相对位置

（12）欧氏距离权值函数dist( )。

大多数神经元网络的输入可通过表达式N=w·X+b 来计算，其中w，b 分别为权矢量和偏差矢量。但有一些神经元的输入可由函数dist()来计算，dist()函数是一个欧氏（Euclidean）距离权值函数，它对输入进行加权，得到被加权的输入。

一般两个向量x和y之间的欧氏（Euclidean）距离D定义为：D=sun((x-y).^2).^0.5。函数dist()调用格式如下。

D=dist(W,X)

或D=dist(pos)

例如任意给定输入和权值，利用dist()函数计算欧氏距离，其代码如下所示。

```
w=[1 2 3];
X=[2;2.2;0.4];
d=dist(w,X)
```

结果显示如下。

```
d =
   2.7928
```

（13）Manhattan 距离权值函数mandist( )。

mandist( ) 函数是一个Manhattan 距离权值函数，它对输入进行加权，得到被加权的输入。一般两个向量x和y之间的Manhattan距离d定义为d=sum(abs(x-y))。其调用格式如下。

```
d=mandist(w,X)
d=mandist(pos)
```

例如任意给定输入和权值，利用mandist()函数计算Manhattan距离，其MATLAB代码如下所示。

```
W = rand(4,3);
P = rand(3,1);
Z = mandist(W,P)
pos = rand(3,10);
D = mandist(pos)
```

得到结果如下。

```
Z =
   1.1715
   1.2536
   1.1152
   0.9667
```

D=

0　0.6904　1.5209　1.2760　0.7674　1.0159　0.6139　0.7919
0.7128　0.6667

0.6904　0　1.9395　1.0512　1.2190　1.3597　0.8936　1.0150
1.0962　1.0853

1.5209　1.9395　　0　1.2656　1.1267　0.7054　1.0459　1.1205
0.8433　0.8897

1.2760　1.0512　1.2656　　0　1.1686　0.5602　1.1019　1.6534
1.1154　1.6816

0.7674　1.2190　1.1267　1.1686　　0　0.8886　0.7124　1.4427
0.6222　0.8507

1.0159　1.3597　0.7054　0.5602　0.8886　　0　0.6059　1.3361
0.5552　1.1214

0.6139　0.8936　1.0459　1.1019　0.7124　0.6059　　0　0.7302
0.2026　0.5797

0.7919　1.0150　1.1205　1.6534　1.4427　1.3361　0.7302　　0
0.8205　0.5920

0.7128　1.0962　0.8433　1.1154　0.6222　0.5552　0.2026
0.8205　　0　0.5662

0.6667　1.0853　0.8897　1.6816　0.8507　1.1214　0.5797
0.5920　0.5662　　0

（14）中点权值初始化函数midpoint( )。

如果竞争层和自组织网络的初始权值选择在输入空间的中间区，利用该函数初始化权值会更加有效。该函数调用格式如下。

w=midpoint(S,Xr)

例如利用函数midpoint初始化权值，MATLAB代码如下。

>>Xr=[0 1;-2 2];

```
>>w=midpoint(4,Xr)
```

结果显示如下。

```
w =
    0.5000         0
    0.5000         0
    0.5000         0
    0.5000         0
```

（15）对输入矢量进行加权函数negdist( )。

函数negdist( )调用格式如下。

```
d=negdist(w,X)
```

例如利用函数negdist对输入向量进行加权操作，MATLAB代码如下。

```
>>W = rand(4,3);
>>P = rand(3,1);
>>Z = negdist(W,P)
```

结果显示如下。

```
Z =
    -0.6733
    -0.7092
    -0.6455
    -0.9401
```

（16）LVQ神经网络的初始化函数initlvq( )。

利用initlvq(　) 函数可建立一个两层（一个竞争层和一个输出层）LVQ神经网络。其调用格式如下。

```
[W1,W2]=initlvq(X,S1,S2)
```

或 [W1,W2]=initlvq(X,S1,T)

（17）用不同的颜色画矢量函数plotvec( )。

函数plotvec(X,x,m)包含一个列矢量矩阵X，标记颜色的行矢量c和一个图形标志m。X的每个列矢量用图形标志画图。每列矢量X(:,i)的数据颜色为c(i)。如果m缺省，则用缺省图形标志“+”。

例如利用函数plotvec对不同的颜色画矢量，MATLAB代码如下。

```
x = [0 1 0.5 0.7;-1 2 0.5 0.1];
c = [1 2 3 4];
plotvec(x,c)
```

得到不同颜色的矢量图如图10-21所示。

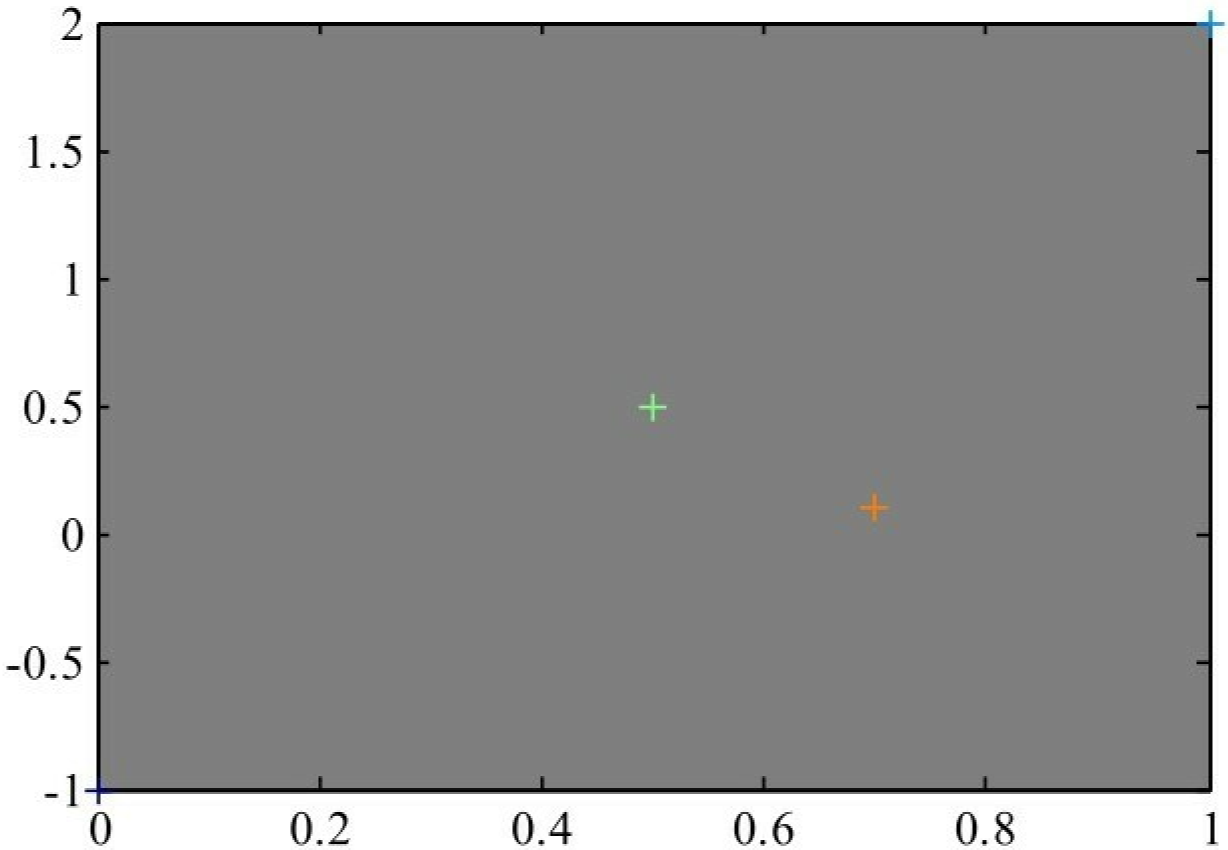


图10-21 不同颜色的矢量图

10.6 竞争型神经网络的应用

竞争型神经网络的基本思想是网络竞争层的各个神经元通过竞争来获得对输入模式的响应机会，最后仅有一个神经元称为竞争的胜利者，并将与获胜神经元有关的各连接权值向着更有利于其竞争的方向调整。

下面用几个经典案例介绍竞争型神经网络的应用。

【例10-7】随机设定15个样本，每个样本包含了测量的20个基因特征，其中前半部分样本（7个样本）是训练网络的基因样本，后半部分样本（7个样本）是待检测的样本。现在需要建立一个竞争型神经网络去预测待检测的样本是患病还是正常基因。

解：由于原始数据是15个样本数据，首先将样本分为训练和预测样本，即前7个为训练样本，后7个为预测样本，MATLAB代码如下。

```
clc
clear
%设定样本数据,患病基因为1，正常基因为0
m=15;
n=20;
data=rand(m,n);
for i=1:m
    for j=1:n
        if data(i,j)>0.2
            data(i,j)=1;
        else
            data(i,j)=0;
        end
```

```
    end
end
%设定训练样本数据
P=data(1:m/2,:);
%设定仿真样本数据
i=m/2+1;
T=data(i:m,:);
%样本数据处理
P=P';
T=T';
Q=minmax(P);
%设定网络学习速率
lr=0.4;
%建立网络
net=newc(Q,2,lr)
%初始化神经网络
net=init(net);
net.trainparam.epochs=50;
%训练神经网络
net=train(net,P);
%用训练数据样本仿真网络
a=sim(net,P);
ac=vec2ind(a);
%输入仿真样本观察网络输出：
Y=sim(net,T);
y=vec2ind(Y);
%计算仿真样本的平均预测误差
```

```
W=zeros(1,m/2);
for i=1:m/2
    if y(i)==T(i)
        W(i)=0;
    else
        W(i)=1;
    end
end
cha=sum(W)/(m/2)
```

神经网络训练窗口如图10-22所示。

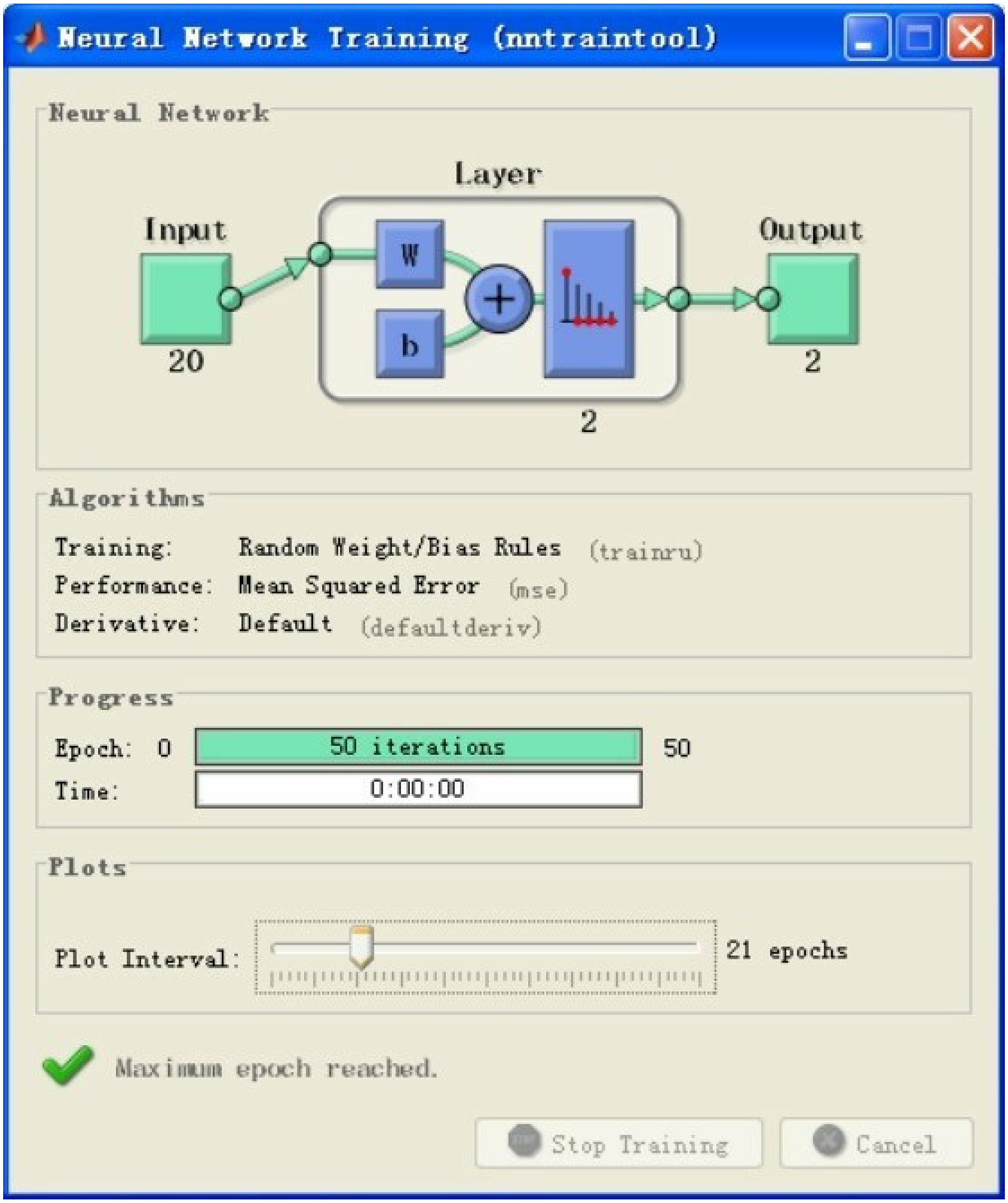


图10-22 神经网络训练窗口

代码运行结果如下。

```
>> y =
   1    1    1    1    1    2    1
>>cha =
   0.1333
```

在 MATLAB 代码中使用了变换函数 vec2ind()，用于将单值向量组变换成下标向量。其调用的格式如下。

ind=vec2ind(vec)

vec：为m行n列的向量矩阵x，x中的每个列向量i，除包含一个1外，其余元素均为0。

ind：为n个元素值为1所在的行下标值构成的一个行向量。

注意：因为题中的输入数据为随机数，所以每次运行后得到的误差不一定相同。如果想得到比较小的误差，需要重复运行代码几次。

【例10-8】随机设定9个数据样本，这些样本分别表示9种不同特性的数据，设计一个SOM网络对不同特性的数据进行分类。

解：根据题意，用函数rand随机设定9个数据样本，然后进行网络的设计及仿真，具体MATLAB代码如下所示。

```
clc
clear
% 设定样本数据
m=9;
n=9;
P=rand(m,n);
for i=1:m
   for j=1:n
      if P(i,j)>0.8
         P(i,j)=1;
      else
         P(i,j)=0;
      end
   end
end
```

```
%转置后符合神经网络的输入格式
P=P';
%newsom建立SOM网络,竞争层为7*7=49个神经元
net=newsom(minmax(P),[7 7]);
plotsom(net.layers{1}.positions)
```

运行上面代码之后得到神经元的位置如图10-23所示。

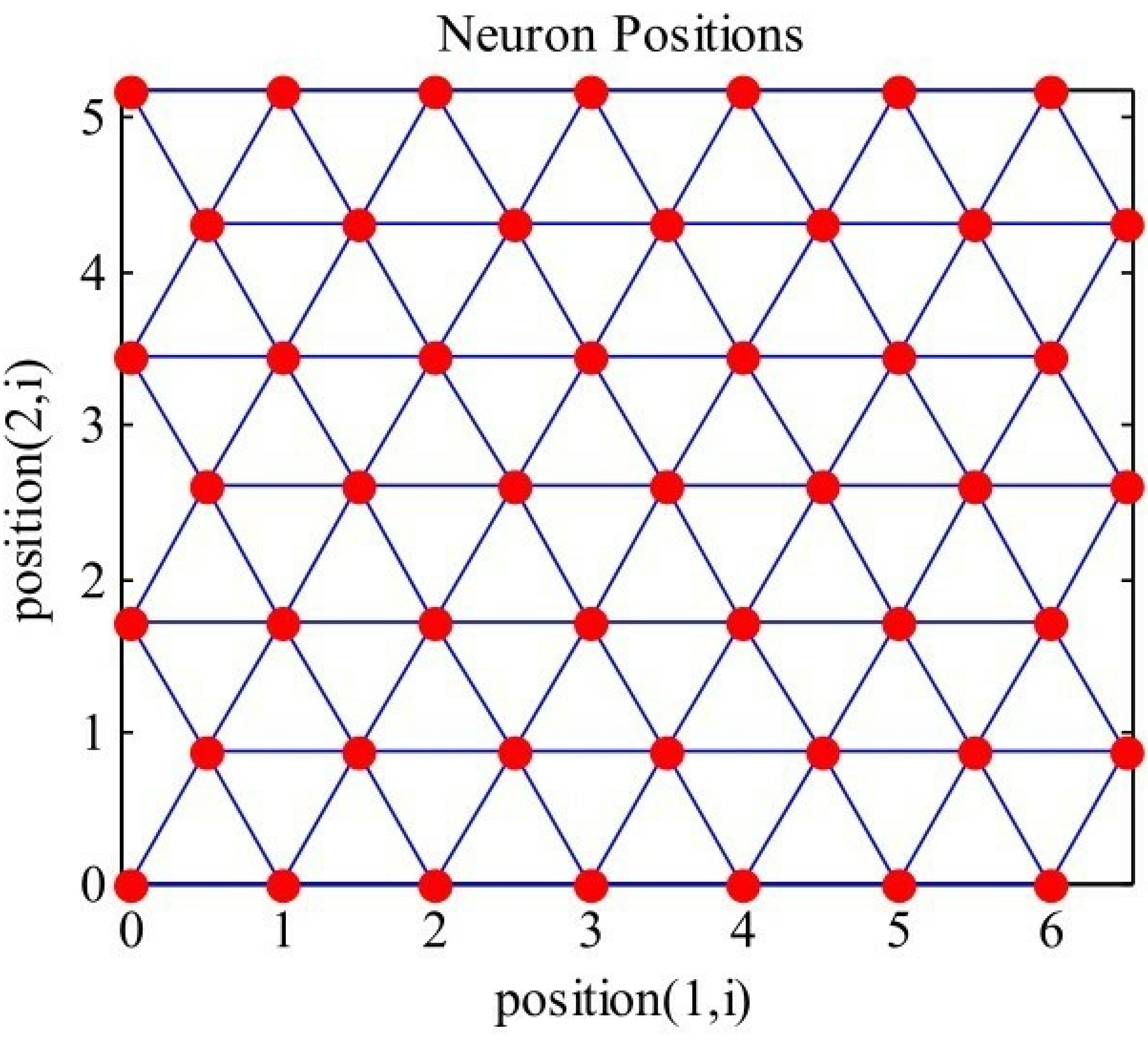


图10-23 神经元的位置示意图

建立好神经网络后，分别设定8种训练的步数，其中训练5、20、40和80步神经网络，其代码如下所示。

```
%5次训练的步数
a=[5 20 40 80 160 320 640 1280];
% 随机初始化一个向量
yc=rands(9,9);
% 训练次数为5次
net.trainparam.epochs=a(1);
%训练网络和查看分类
net=train(net,P);
%仿真网络
y=sim(net,P);
yc(1,:)=vec2ind(y);
figure(1)
subplot(4,2,1);
plotsom(net.IW{1,1},net.layers{1}.distances)
% 训练次数为20次
net.trainparam.epochs=a(2);
% 训练网络和查看分类
net=train(net,P);
%仿真网络
y=sim(net,P);
yc(2,:)=vec2ind(y);
subplot(4,2,2);
plotsom(net.IW{1,1},net.layers{1}.distances)
% 训练次数为40次
net.trainparam.epochs=a(3);
% 训练网络和查看分类
net=train(net,P);
```

```
%仿真网络
y=sim(net,P);
yc(3,:)=vec2ind(y);
subplot(4,2,3);
plotsom(net.IW{1,1},net.layers{1}.distances)
% 训练次数为80次
net.trainparam.epochs=a(4);
% 训练网络和查看分类
net=train(net,P);
%仿真网络
y=sim(net,P);
yc(4,:)=vec2ind(y);
subplot(4,2,4);
plotsom(net.IW{1,1},net.layers{1}.distances)
```

训练5、20、40和80步神经网络，得到权值变化如图10-24所示。

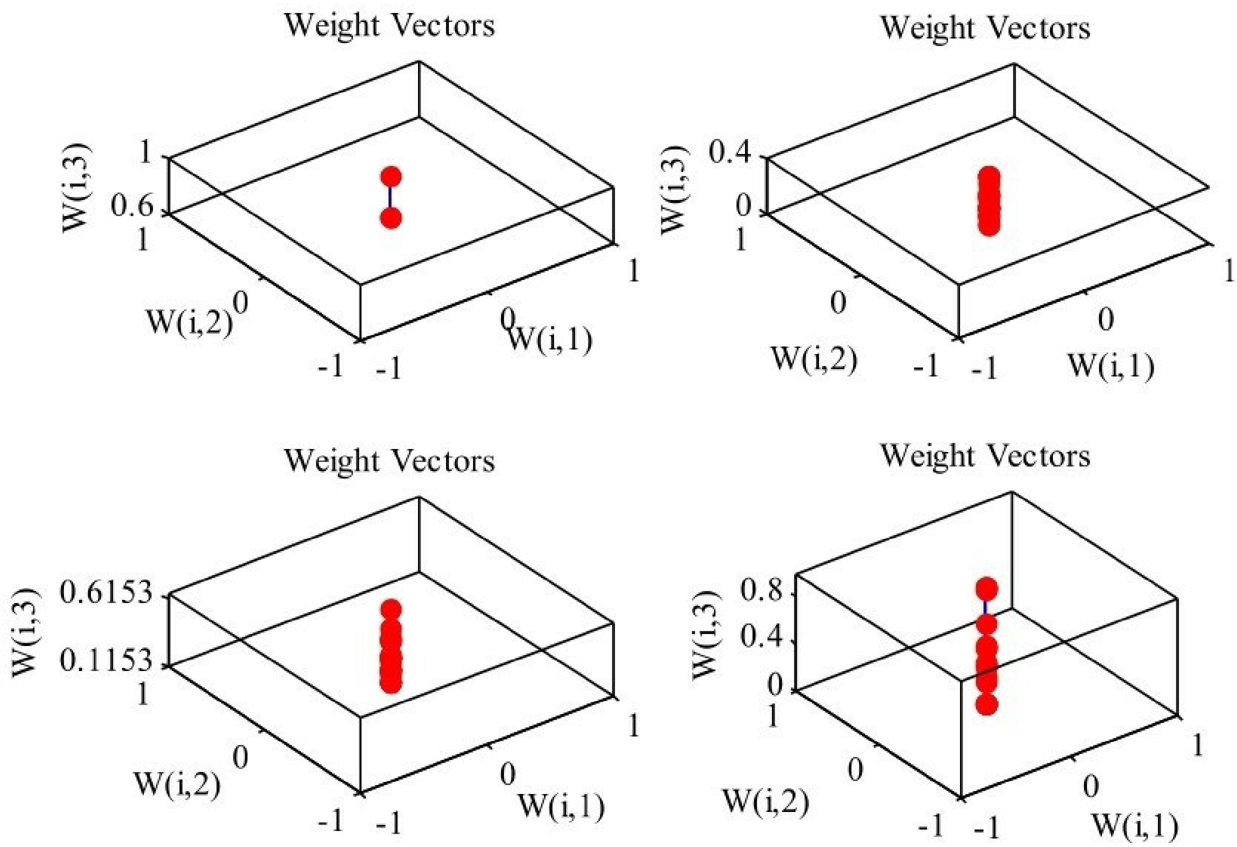


图10-24 不同神经网络训练步数得到权值的变化

分别设定8种训练的步数，其中训练160、320、640和1280步神经网络，其代码如下所示。

```
% 训练次数为160次
net.trainparam.epochs=a(5);
% 训练网络和查看分类
net=train(net,P);
%仿真网络
y=sim(net,P);
yc(5,:)=vec2ind(y);
subplot(4,2,5);
plotsom(net.IW{1,1},net.layers{1}.distances)
% 训练次数为320次
```

```
net.trainparam.epochs=a(6);
% 训练网络和查看分类
net=train(net,P);
%仿真网络
y=sim(net,P);
yc(6,:)=vec2ind(y);
subplot(4,2,6);
plotsom(net.IW{1,1},net.layers{1}.distances)
% 训练次数为640次
net.trainparam.epochs=a(7);
% 训练网络和查看分类
net=train(net,P);
%仿真网络
y=sim(net,P);
yc(7,:)=vec2ind(y);
subplot(4,2,7);
plotsom(net.IW{1,1},net.layers{1}.distances)
% 训练次数为1280次
net.trainparam.epochs=a(8);
% 训练网络和查看分类
net=train(net,P);
%仿真网络
y=sim(net,P);
yc(8,:)=vec2ind(y);
subplot(4,2,8);
plotsom(net.IW{1,1},net.layers{1}.distances)
```

训练160、320、640和1280步神经网络，得到权值变化如图10-25所

示。

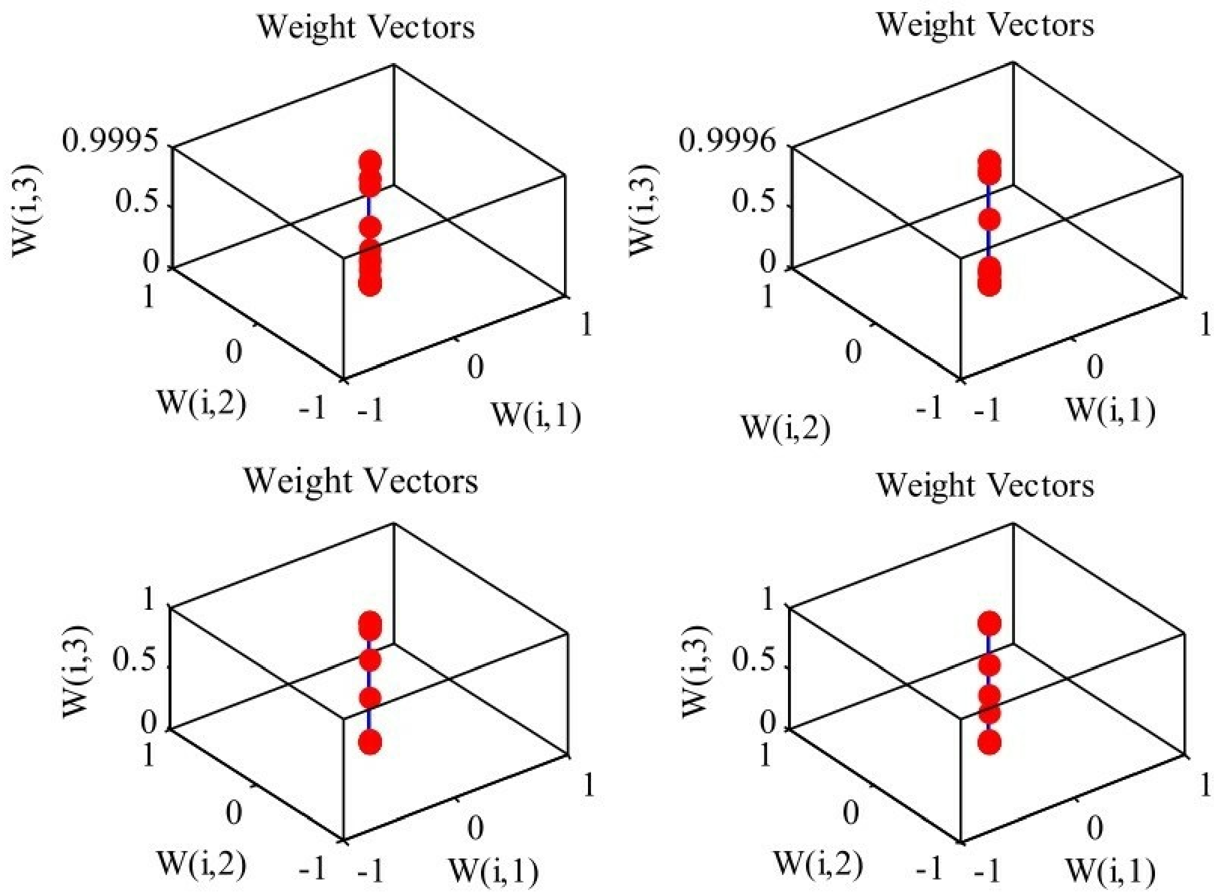


图10-25 不同神经网络训练步数得到权值的变化

其中，训练次数为1280时，用输入样本仿真得到的输出如下。

```
>>yc=
14   43   14   7    5    14   7    14   6
43   28   1    28   43   8    28   43   49
7    21   43   28   1    49   42   29   22
36   42   1    14   43   18   7    25   49
15   28   43   49   1    24   46   4    7
22   42   14   47   1    43   38   10   6
49   1    11   22   27   45   36   32   7
28   22   49   43   7    25   46   4    1
```

-0.620075269571580   0.324641480607133   -   0.988491977227226
0.690399525925539

-0.507479946173335   0.804370923856130   -0.832156003734812
-0.292981041423053

0.475038126465344

由yc的值可以知道，样本1、3、6和样本8的特性相同（值均为14），样本4和样本7的特性相同（值均为7）。

训练完成网络后，设定测试样本仿真网络。

```
%网络作分类的预测
%测试样本输入
t=[0.8712 1.0000 0.1858 -0.4915 0.4918 0.7111 0.5345 0.7941 0.5941]';
%网络仿真
r=sim(net,t);
%将单值向量转变成下标向量。
rr=vec2ind(r)
%查看网络拓扑学结构
figure(2)
plotsomtop(net)
```

网络拓扑学结构如图10-26所示，竞争层神经元有7×7=49个。

```
%查看临近神经元直接的距离情况
figure(3)
plotsomnd(net)
```

临近神经元直接的距离如图 10-27 所示，图中，蓝色表示神经元，红色线表示神经元直接的连接，每个菱形中的颜色表示神经元之间距离的远近，从黄色到黑色，颜色越深说明神经元之间的距离越远。

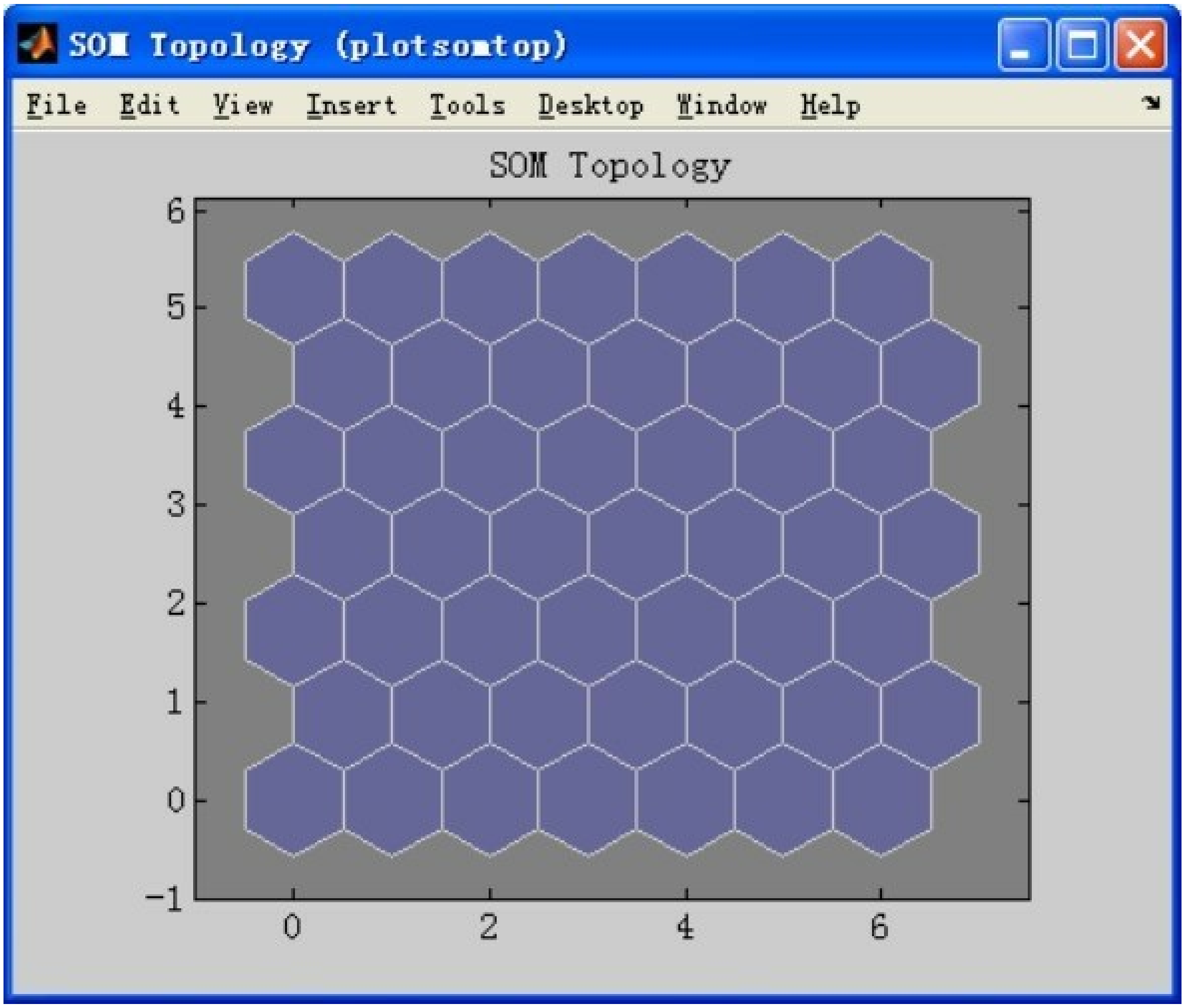


图10-26 网络拓扑学结构

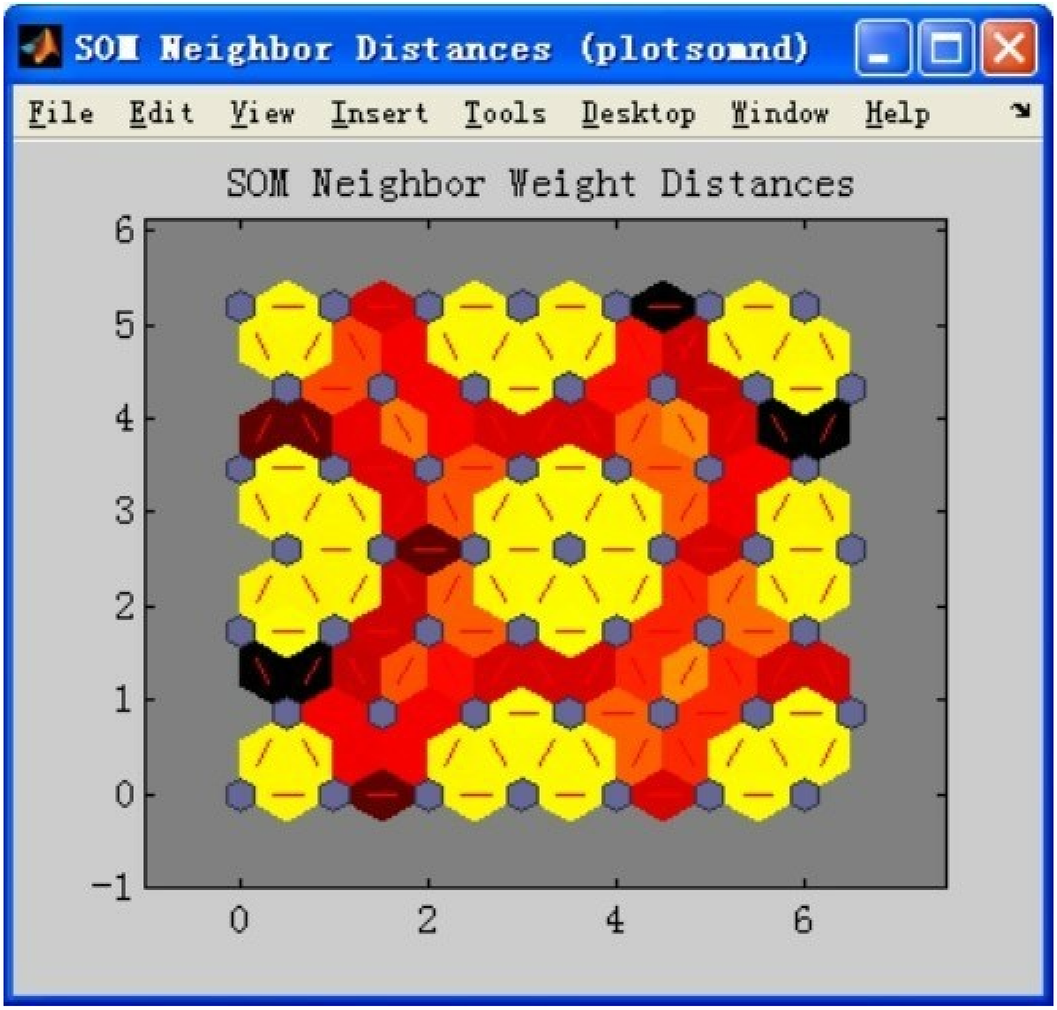


图10-27 临近神经元直接的距离

%查看每个神经元的分类情况

figure(4)

plotsomhits(net,P)

每个神经元的分类情况如图10-28所示，图中蓝色神经元表示竞争胜利的神经元。

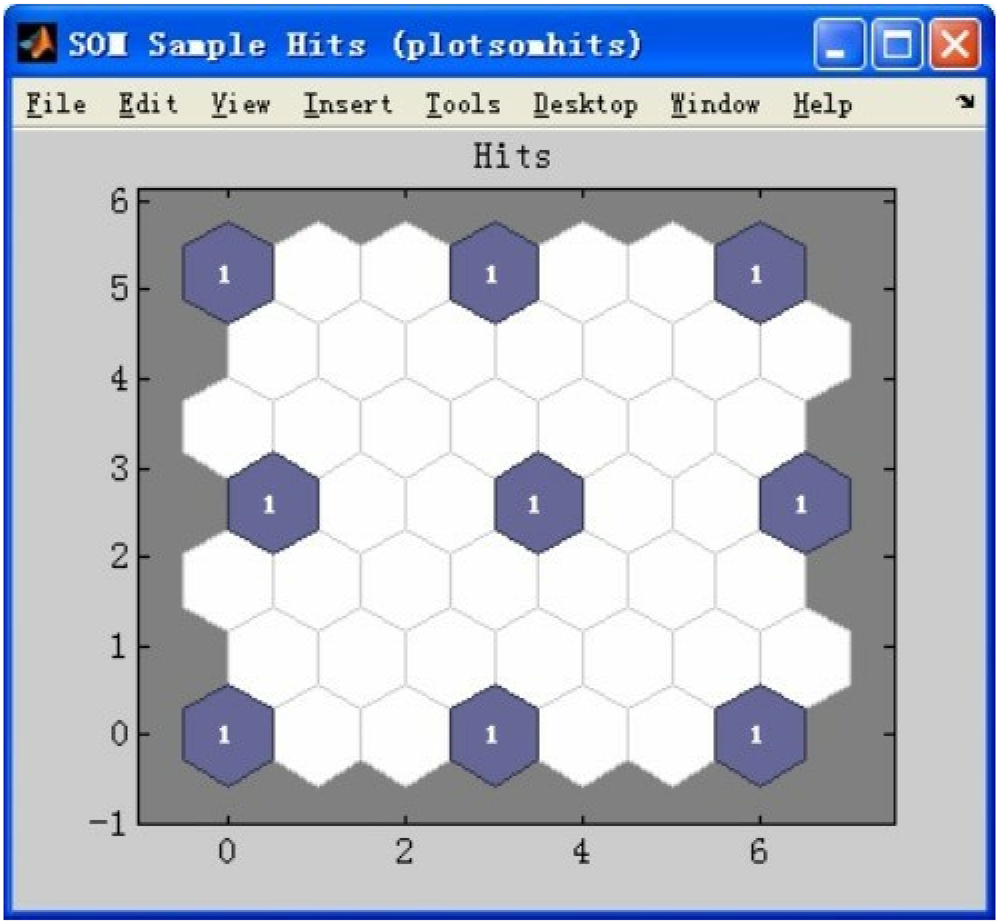


图10-28 每个神经元的分类情况

10.7 本章小结

本章主要介绍了竞争型神经网络，包括自组织型神经网络、自组织特征映射神经网络、自适应共振理论神经网络。除此之外，还介绍了竞争型神经网络工具箱中比较重要的几个函数的应用。在本章的最后，举例说明了竞争型神经网络如何应用到实际中。

第11章 神经网络的Simulink应用

Simulink是MATLAB最重要的组件之一，也被称为MATLAB的系统动态仿真工具箱。利用这个工具箱，可以对系统进行建模、仿真和综合分析等操作。Simulink 神经网络仿真模型库是Simulink模块库的重要部件之一。

本章将重点介绍Simulink神经网络仿真模块的概念及其使用方法。

学习目标：

·熟悉神经网络仿真模块

·掌握神经网络仿真模块的使用

11.1 基于Simulink的神经网络模块

神经网络工具箱中提供了一套可在 Simulink 中用来建立神经网络的模块，对于在MATLAB 工作空间中建立的网络，也能够使用函数gensim()生成一个相应的Simulink 网络模块。下面介绍神经网络simulink模块及其简单使用方法。

11.1.1 神经网络模块

在Simulink库浏览窗口的Neural Network Toolbox 节点上，通过单击鼠标右键后，便可打开图11-1 所示的Neural Network Toolbox 模块集窗口。

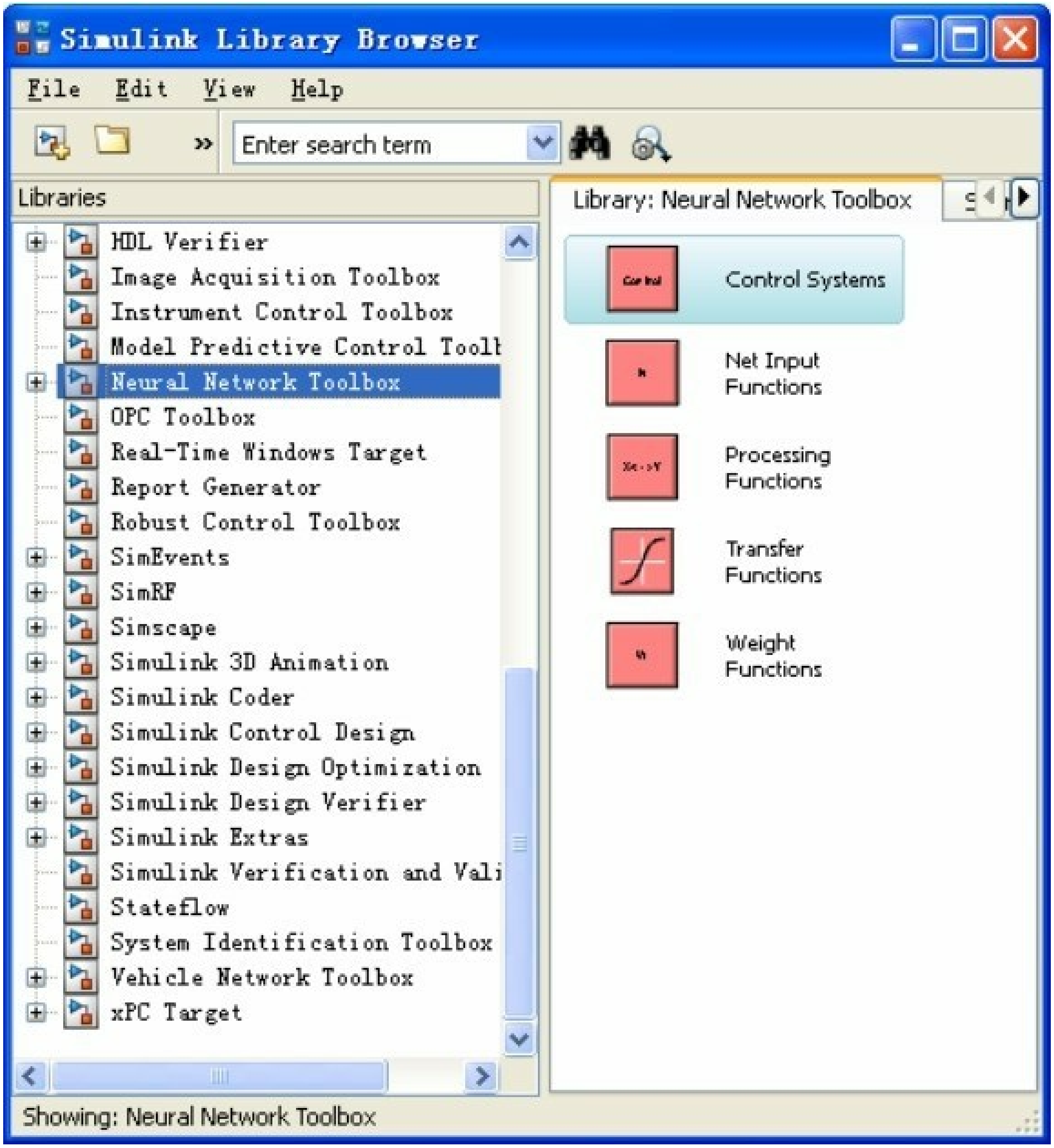


图11-1 Neural Network Toolbox模块集窗口

在Neural Network Toolbox 模块集中包含了5 个模块库，用鼠标左键双击各个模块库的图标，便可打开相应的模块库。

1.Control Systems（控制系统模块库）

用鼠标左键单击Control Systems模块库的图标，便可打开图11-2 所示的控制系统模块库窗口。图11-2中的神经网络控制系统模块库中包含3个控制器和1个示波器。

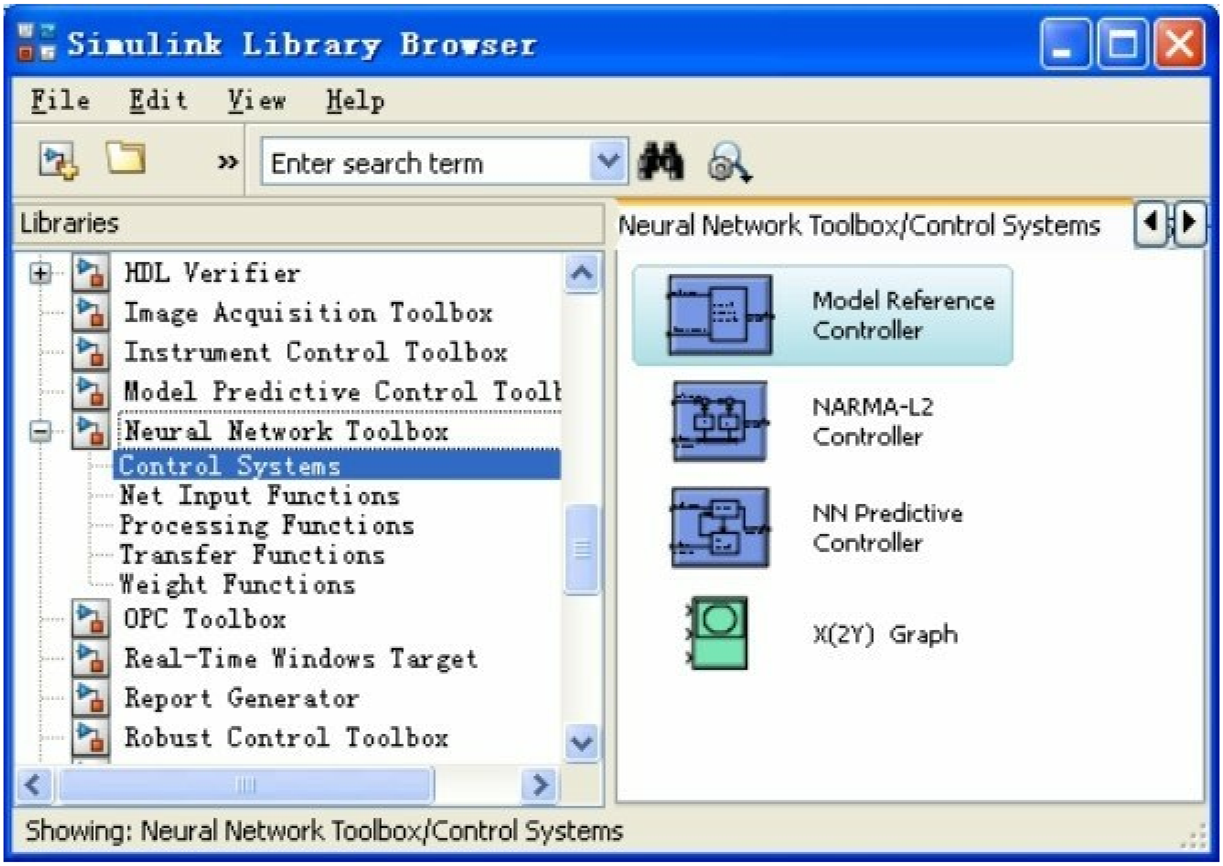


图11-2 控制系统模块库窗口

其中，Model Reference Controller（模型参考控制器）的设置窗口如图11-3 所示。

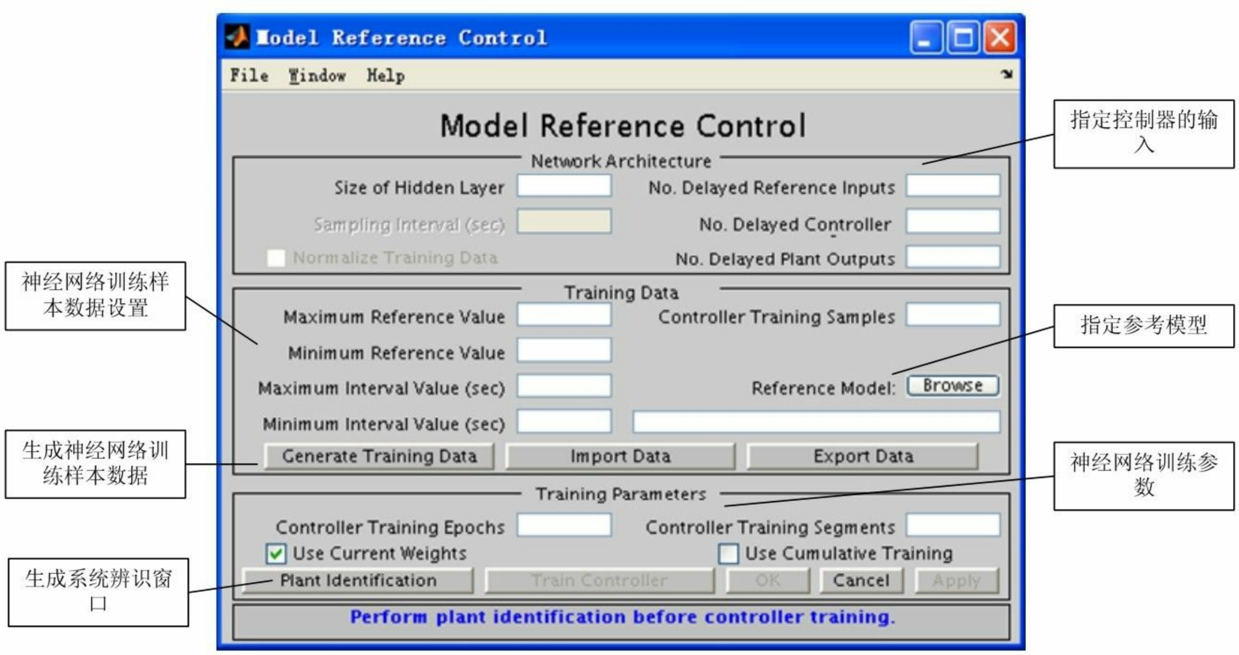


图11-3 Model Reference Controller的设置窗口

NARMA-L2 Controller（反馈线性化控制器）的设置窗口如图11-4所示。

NN Predictive Controller（神经网络模型预测控制）的设置窗口如图11-5所示。

2.Net Input Functions（网络输入模块库）

用鼠标左键单击Net Input Functions模块库的图标，便可打开图11-6所示的网络输入模块库窗口。

网络输入模块库中的每一个模块都能够接受任意数目的加权输入向量、加权的层输出向量，以及偏值向量，并且返回一个网络输入向量。

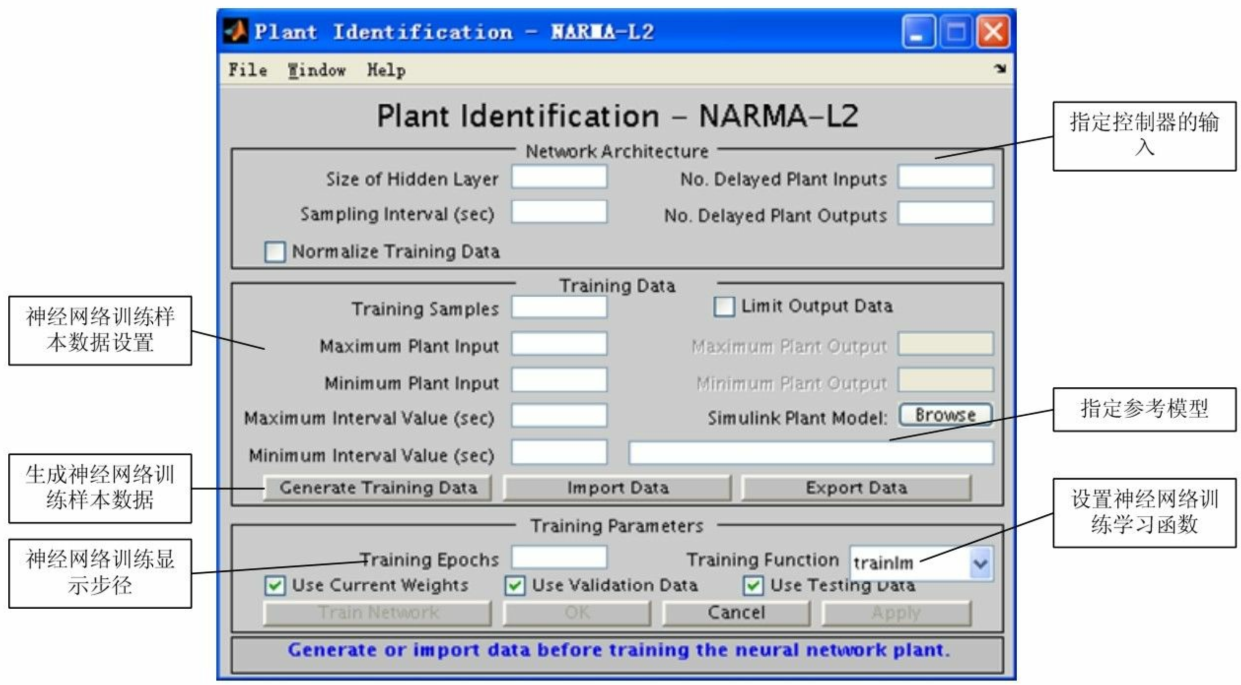


图11-4 NARMA-L2 Controller的设置窗口

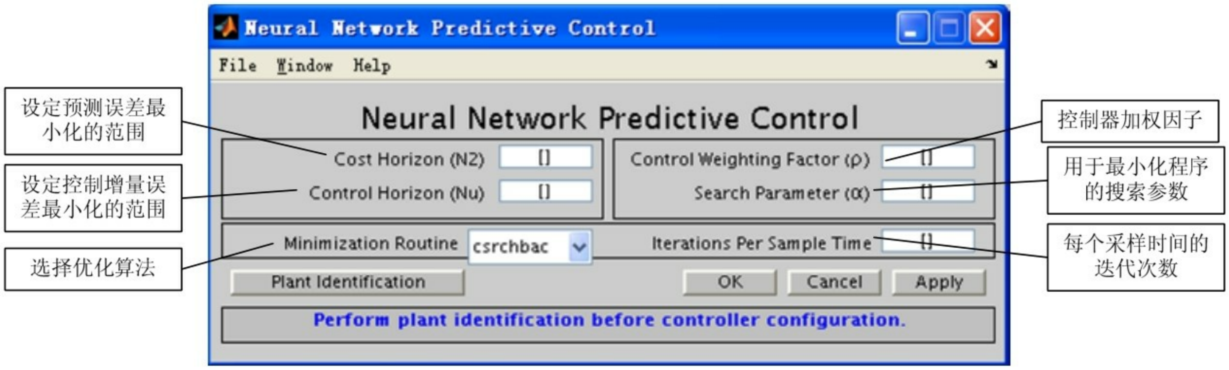


图11-5 NN Predictive Controller 的设置窗口

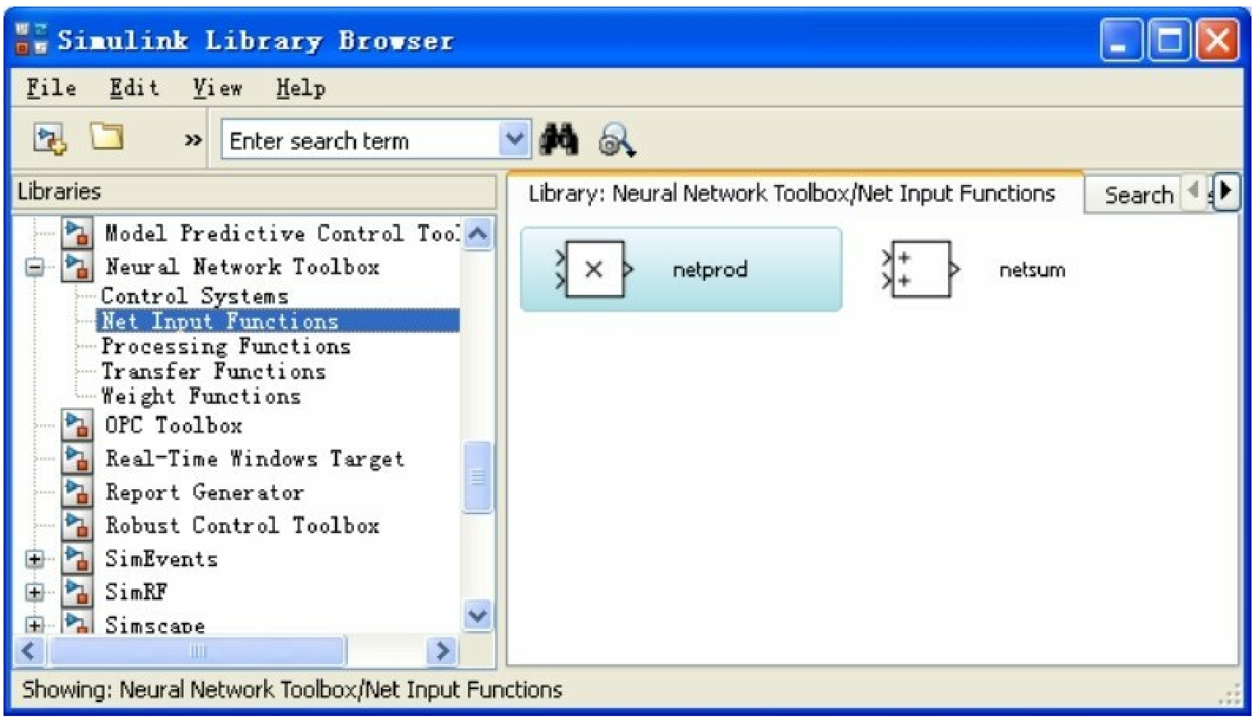


图11-6 Net Input Functions窗口

3.Processing Functions（过程函数模块库）

用鼠标左键单击Processing Functions模块库的图标，便可打开图11-7 所示的过程函数模块库窗口。

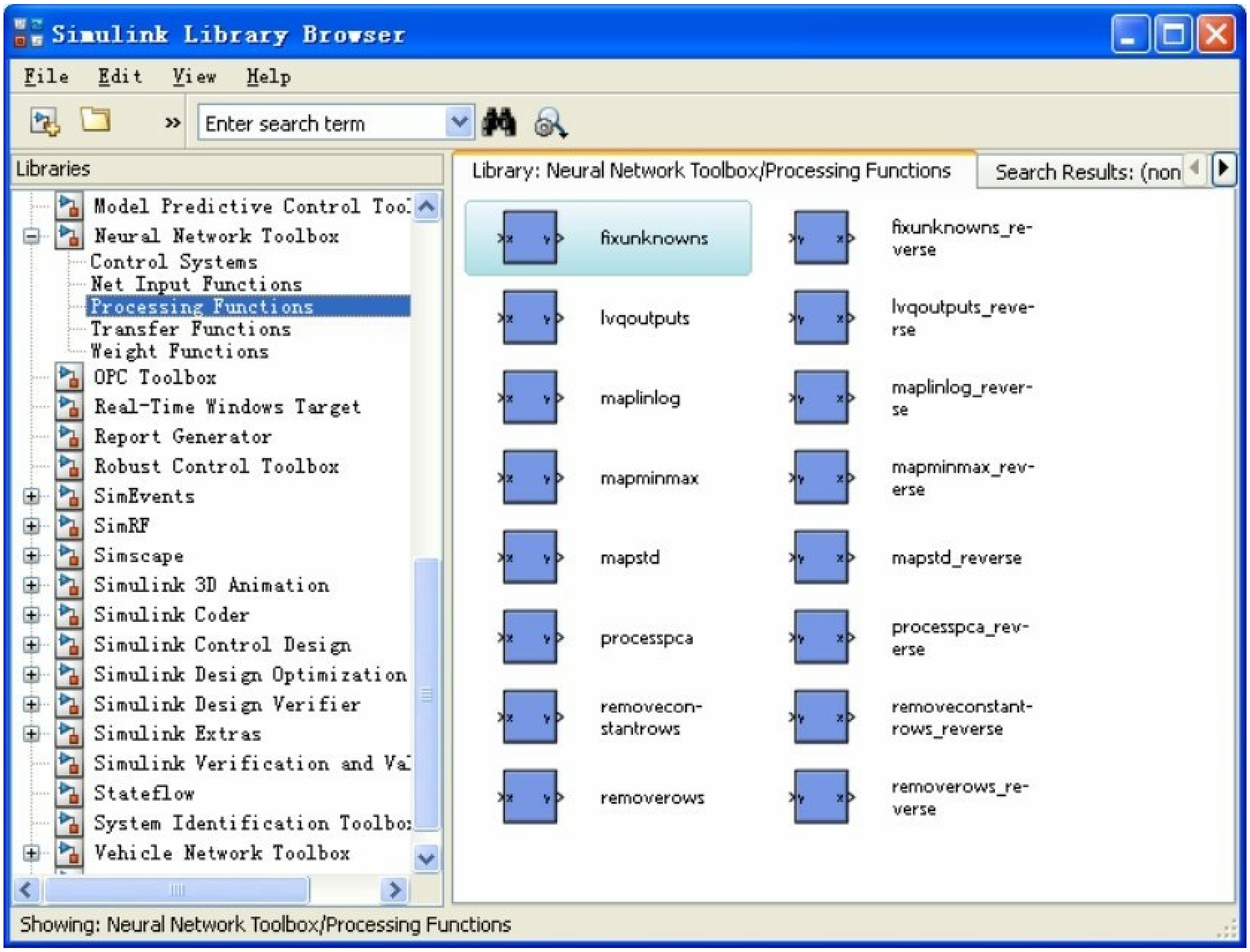


图11-7 Processing Functions窗口

4.Transfer Functions（传输函数模块库）

用鼠标左键单击Transfer Functions模块库的图标，便可打开图11-8所示的传输函数模块库窗口。

在图11-8所示窗口中，神经网络建模用到最多的模块是hardlims、purelin和tansig。下面举例说明这3个模块的应用。

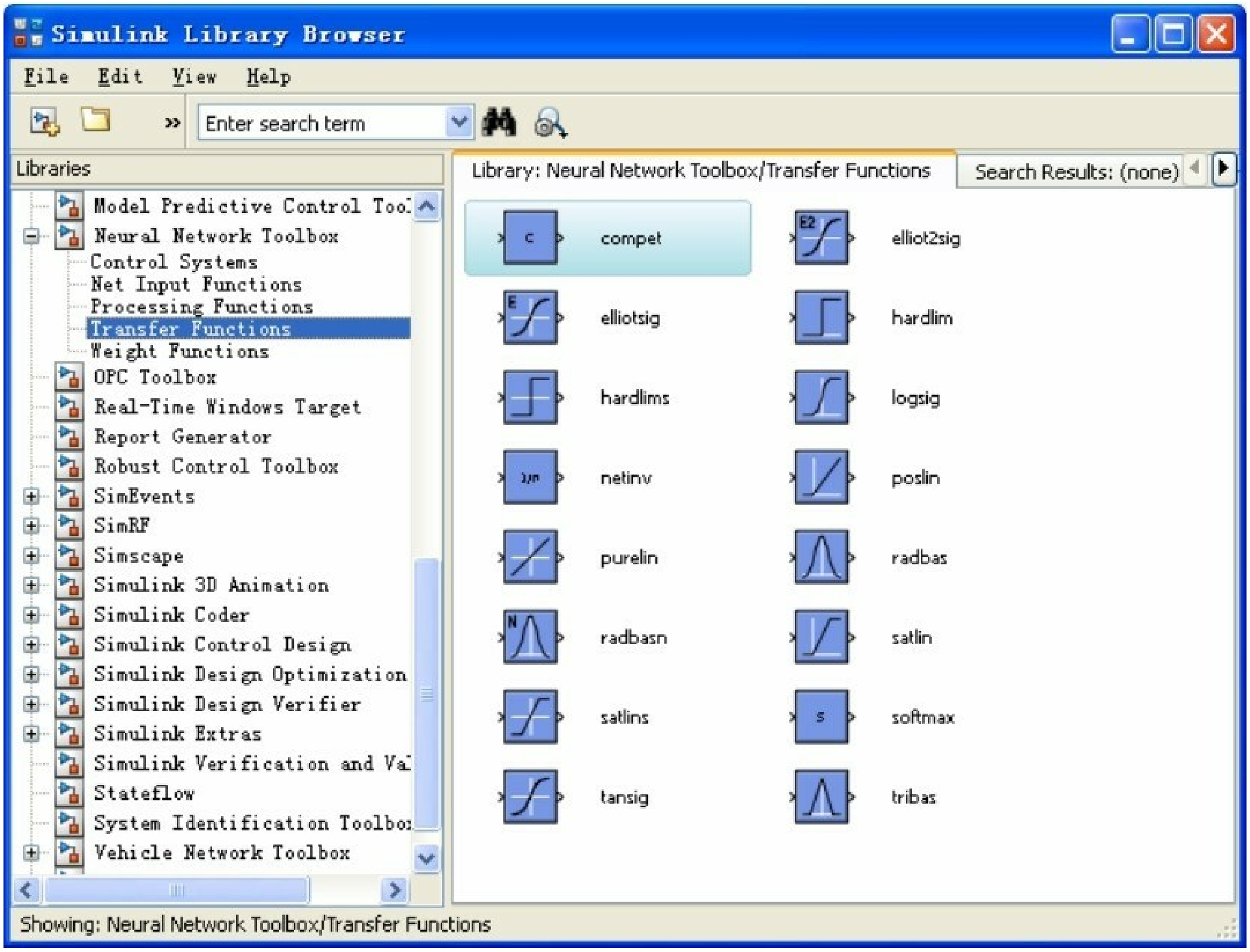


图11-8 Transfer Functions窗口

【例11-1】用Transfer Functions 中的hardlims、purelin 和tansig 模块搭建模型，根据指定的输入变量，得到输出图像。同时，在MATLAB中通过编程实现上述模型的功能。

解：设定输入变量为 n=[-5　-4　-3　-2　-1　0　1　2　3　4　5]。

（1）利用hardlims模块建立模型如图11-9所示。

运行模型得到结果曲线如图11-10所示。从图中可以看出，当输入小于0的时候，输出为−1；当输入大于0的时候，输出为1。

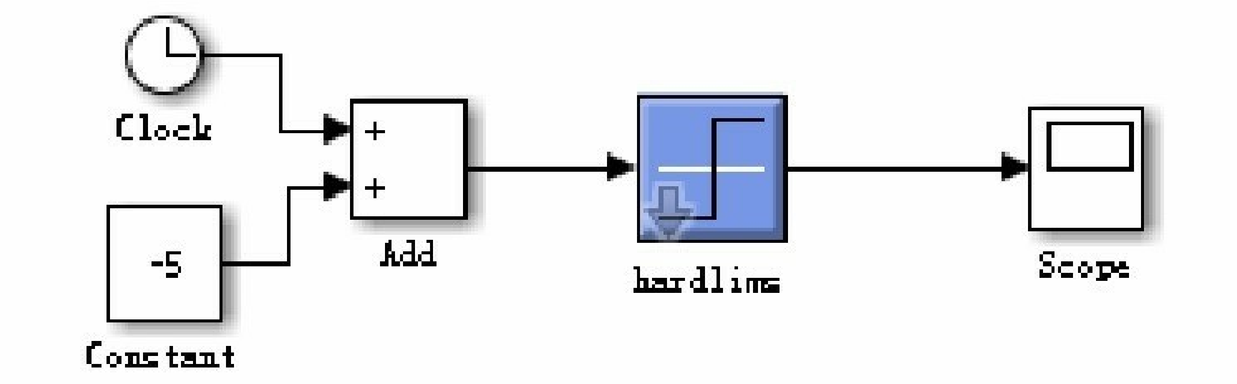


图11-9 利用hardlims模块建立的模型

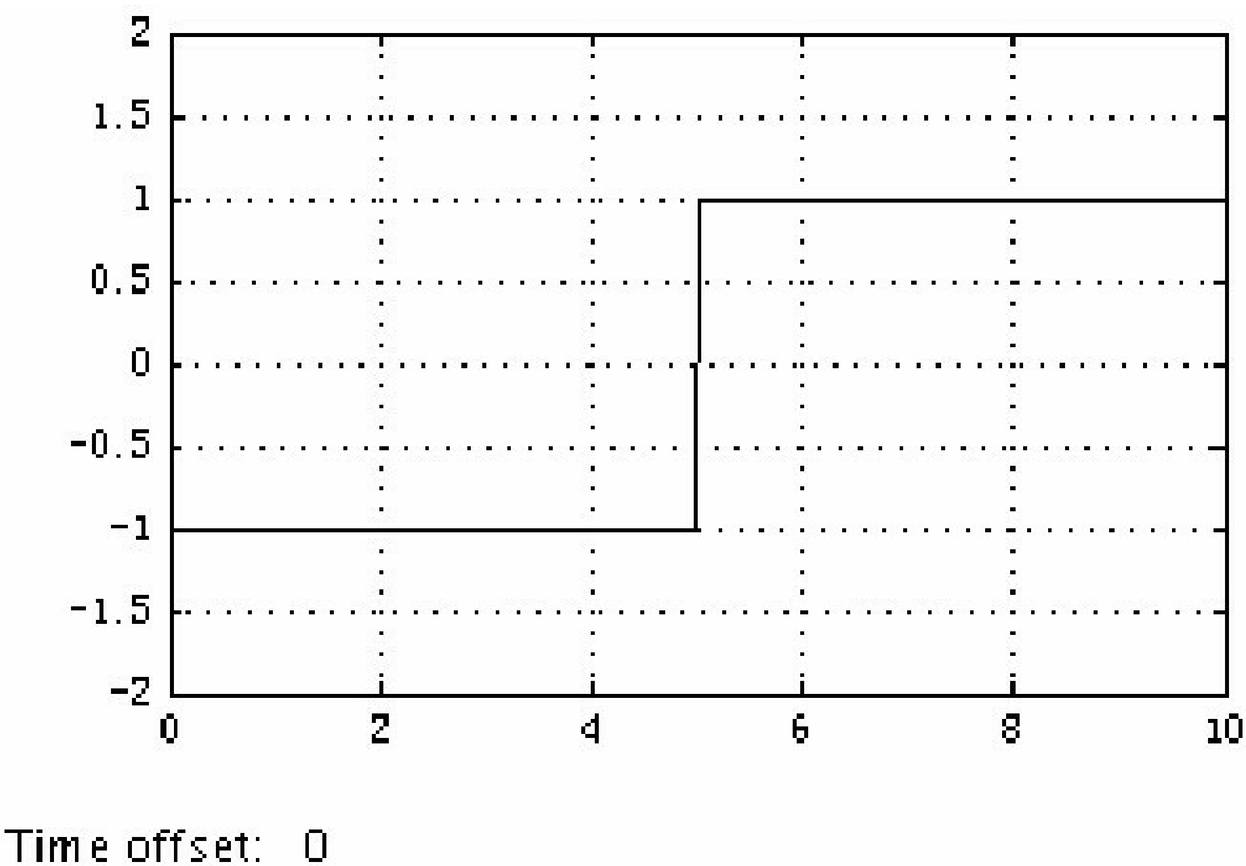


图11-10 模型运行结果曲线

除了搭建模型，通过函数也可以实现hardlims模块的功能，具体程序如下。

```
clc
clear
n =-5:1:5;
```

```
a = hardlims(n);
plot(n,a)
```

得到结果如图11-11所示。

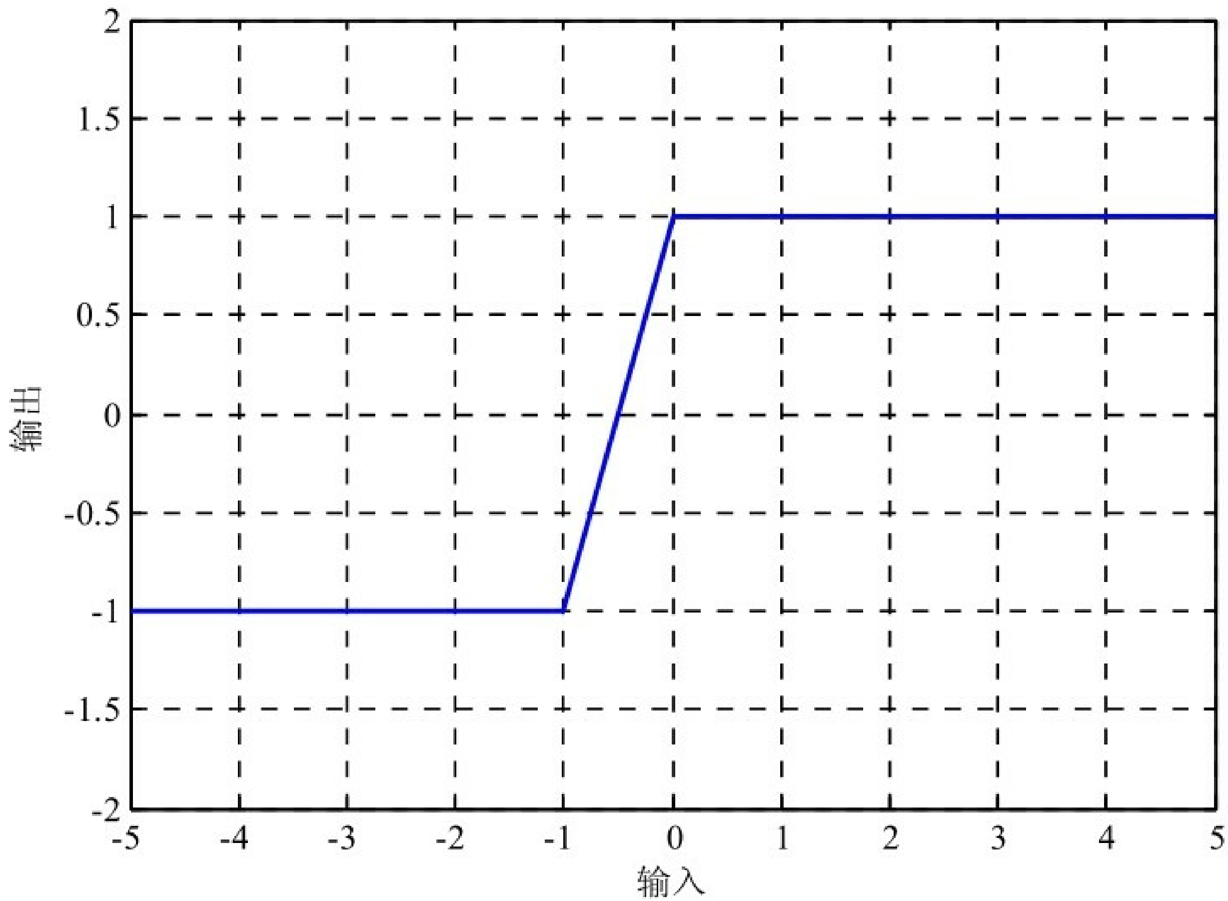


图11-11 函数hardlims运行得到的结果

从图11-10和图11-11可以得到，用hardlims模块和hardlims函数得到的结果一致，函数和模块可以实现相同的功能。

（2）利用purelin模块建立模型如图11-12所示。

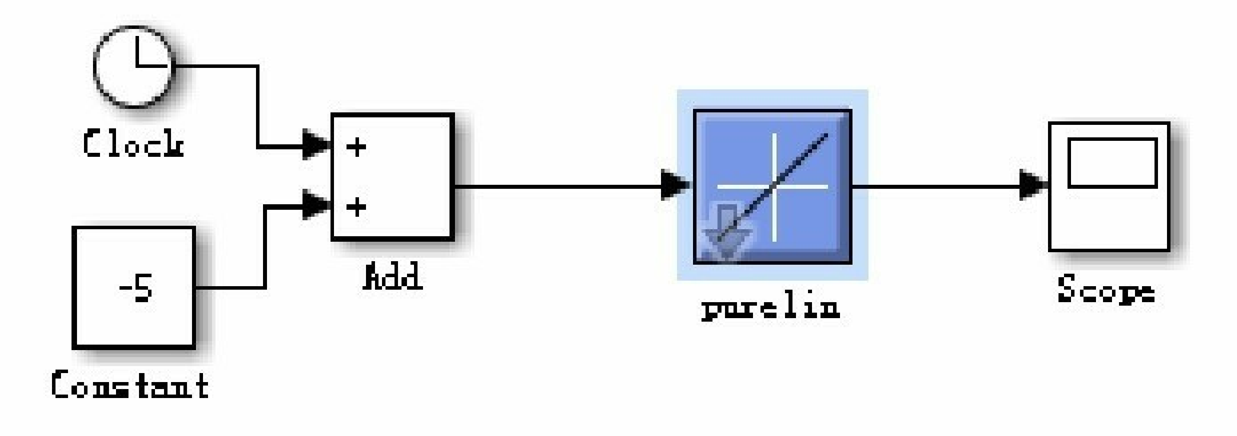


图11-12 利用purelin模块建立的模型

运行模型得到结果曲线如图11-13所示。从图中可以看出，输入与输出值一致。

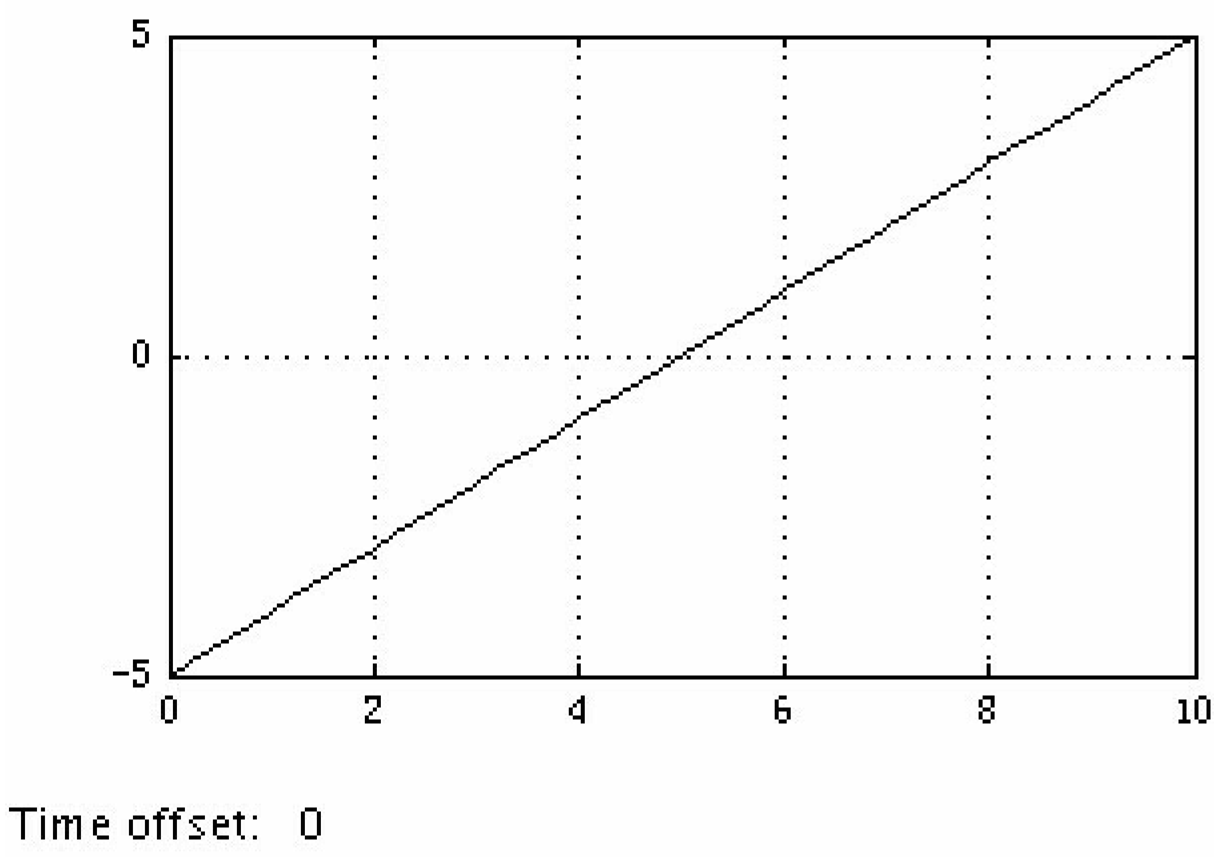


图11-13 模型运行结果曲线

除了搭建模型，通过函数也可以实现purelin模块的功能，具体程序如下。

```
clc
clear
n =-5:1:5;
a = purelin(n);
plot(n,a)
```

得到结果如图11-14所示。

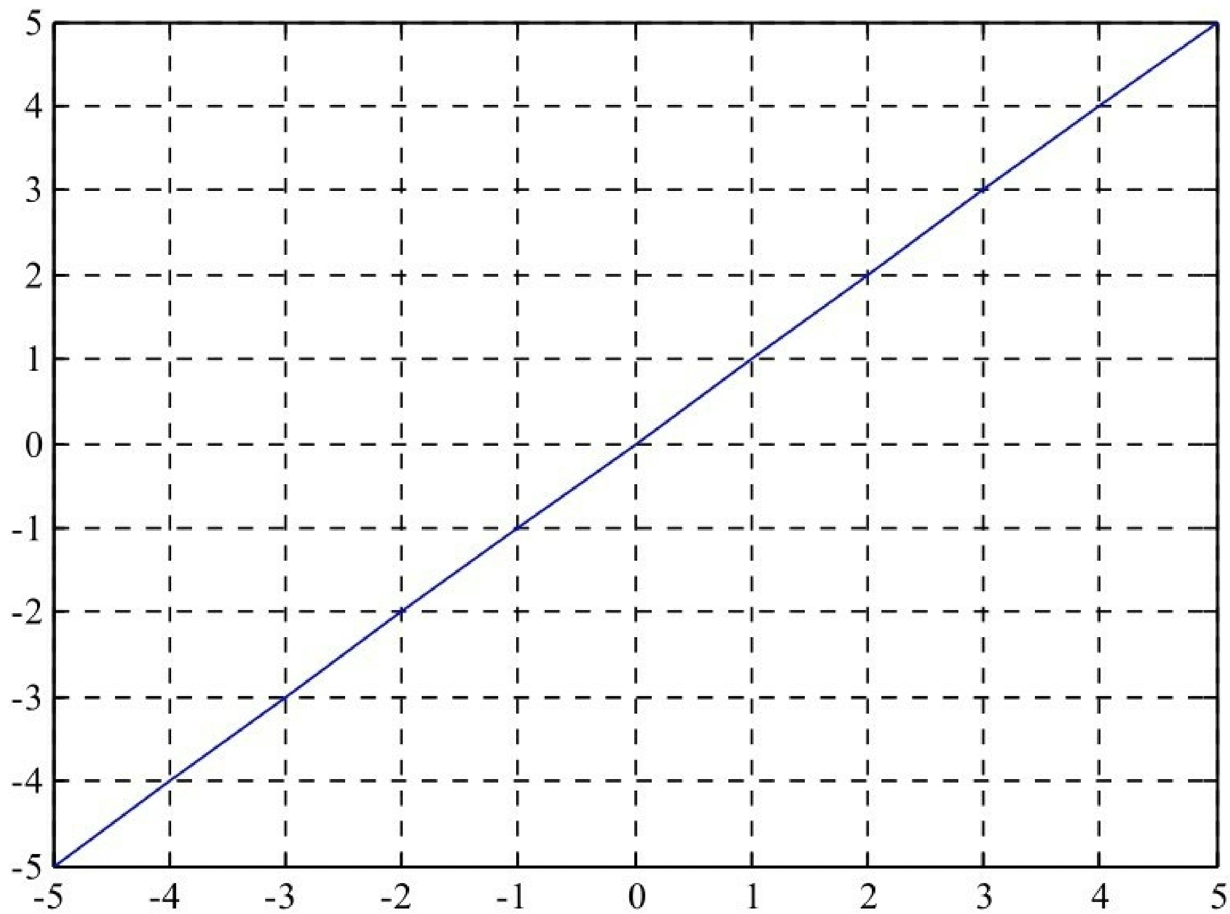


图11-14 函数purelin运行得到的结果

从图11-13和图11-14可以得到，用purelin模块和purelin函数得到的结果一致，函数和模块可以实现相同的功能。

（3）利用tansig模块建立模型如图11-15所示。

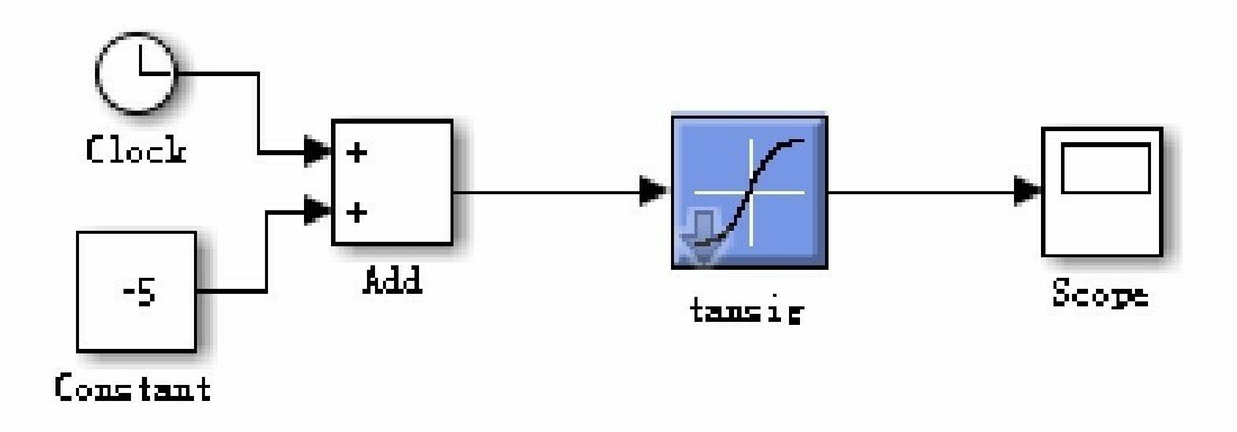


图11-15 利用tansig模块建立的模型

运行模型得到结果曲线如图11-16所示。从图中可以看出，输入与输出值一致。

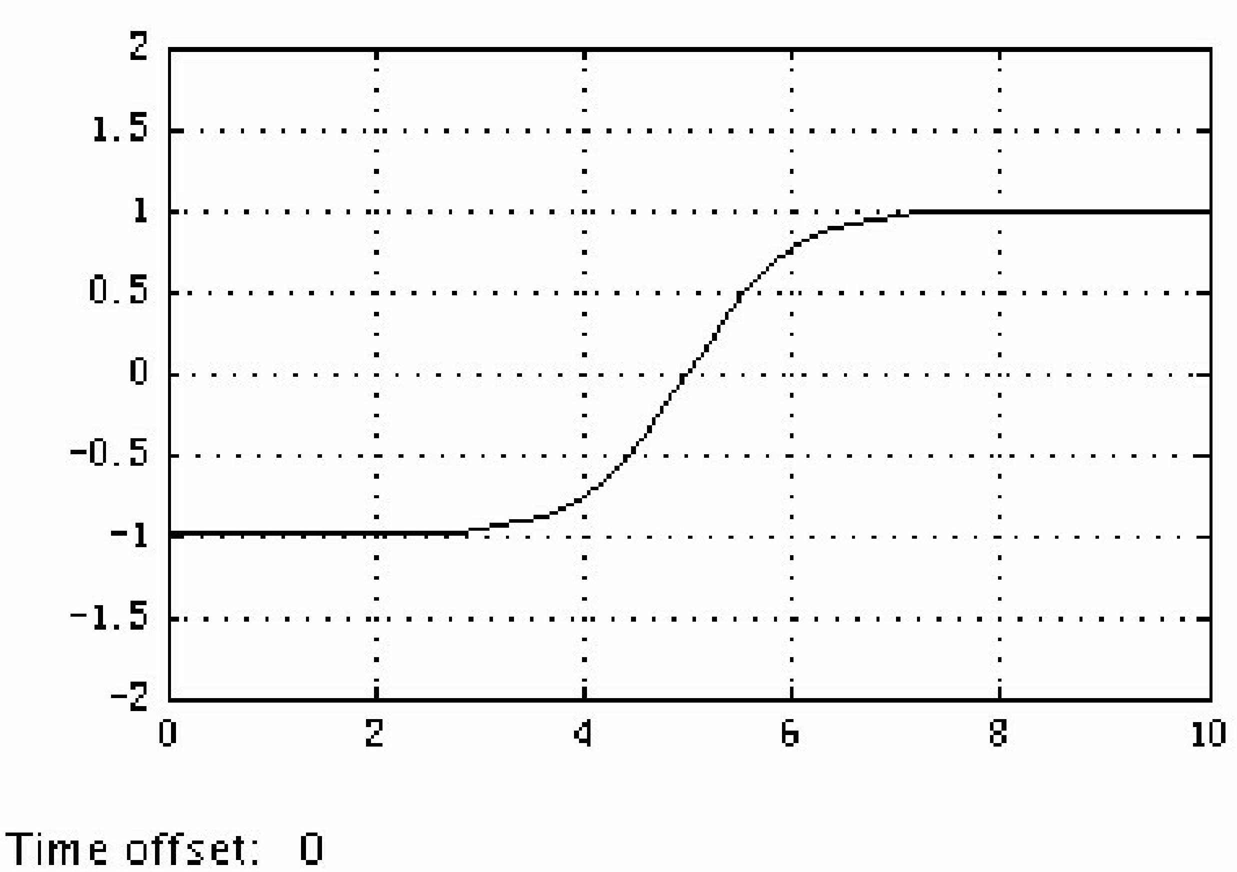


图11-16 模型运行结果曲线

除了搭建模型，通过函数也可以实现tansig模块的功能，具体程序如下。

```
clc
clear
n =-5:1:5;
a = tansig (n);
plot(n,a)
```

得到结果如图11-17所示。

从图11-16和图11-17可以得到，用tansig模块和tansig函数得到的结果一致，函数和模块可以实现相同的功能。

5.Weight Functions（权值模块库）

用鼠标左键双击Weight Functions 模块库的图标，便可打开图11-18所示的权值模块库窗口。权值模块库中的每个模块都以一个神经元权值向量作为输入，并将其与一个输入向量（或者是某一层的输出向量）进行运算，得到神经元的加权输入值。

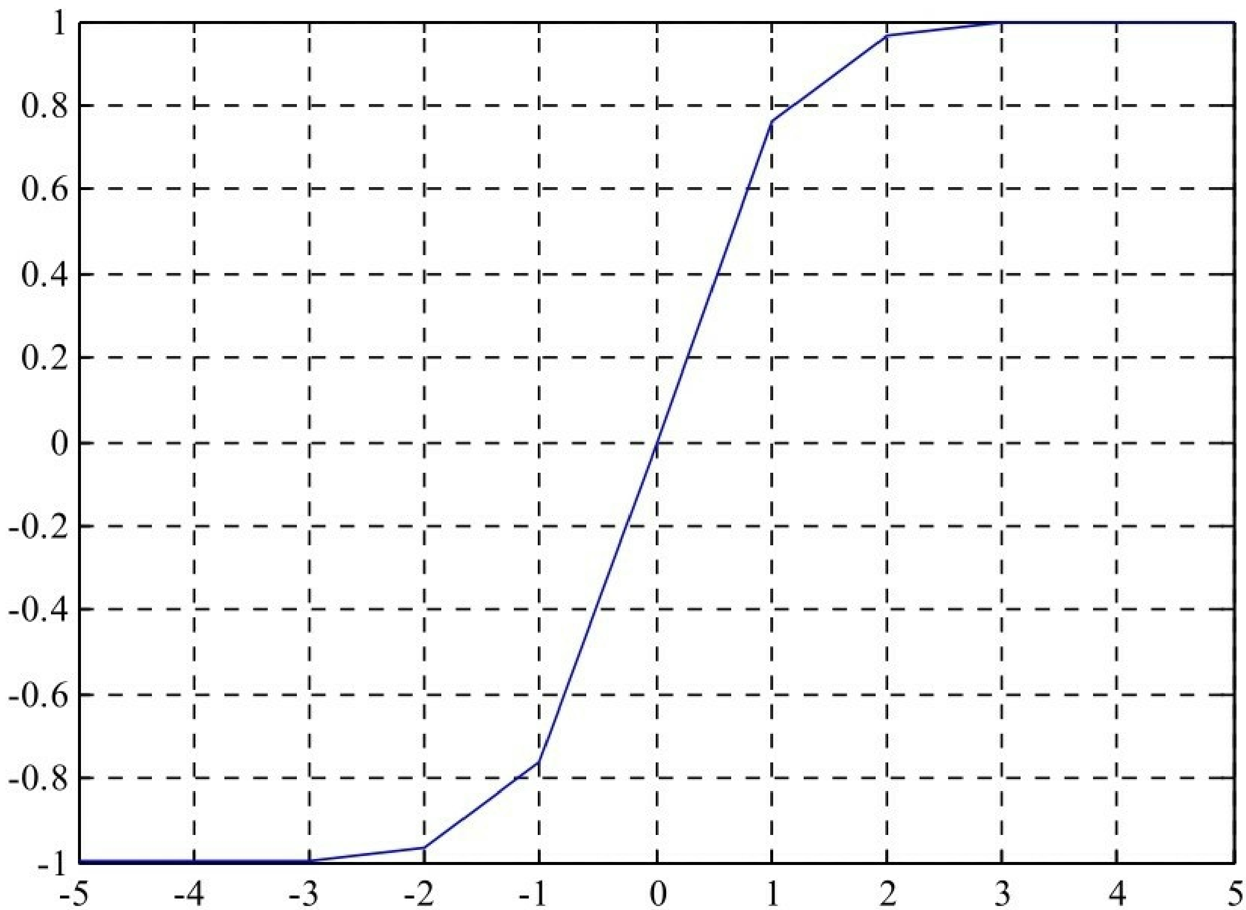


图11-17 函数tansig运行得到的结果

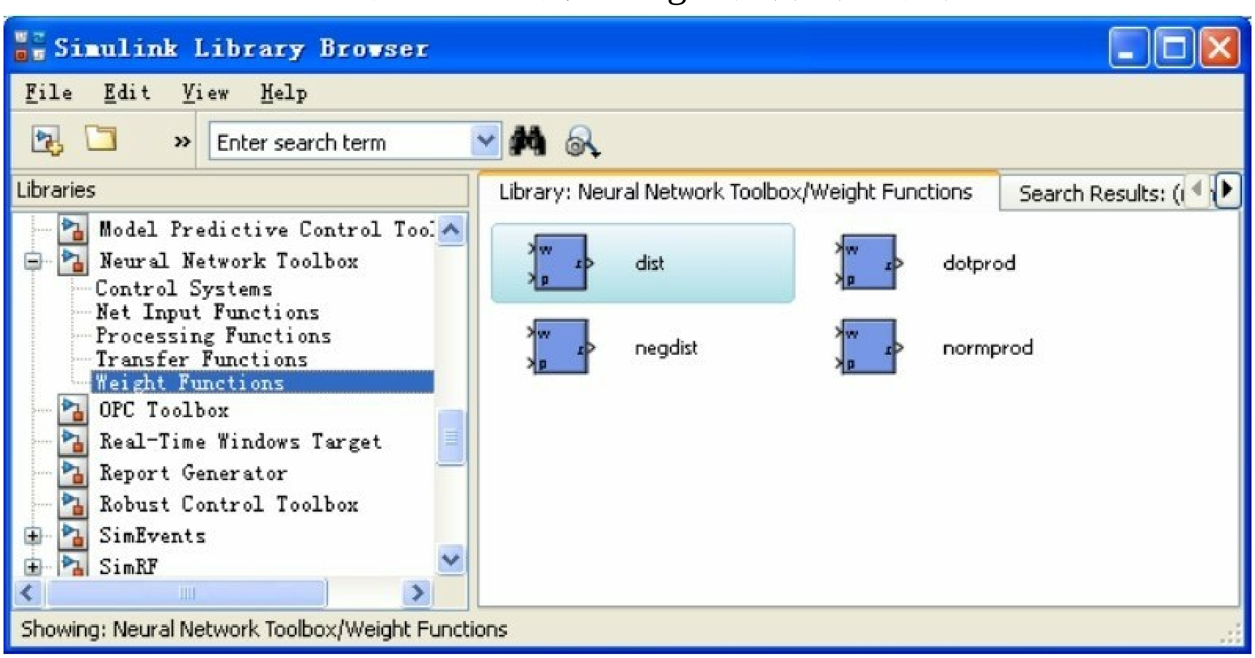


图11-18 权值模块库窗口

上面的这些模块需要的权值向量必须定义为列向量。这是因为Simulink中的信号可以为列向量，但是不能为矩阵或者行向量。

11.1.2 模块的生成

在MATLAB工作空间中，利用函数gensim()，能够对一个神经网络生成其模块化描述，从而可在Simulink 中对其进行仿真。gensim( )函数的调用格式如下。

gensim(net,st)

其中第一个参数指定了MATLAB工作空间中需要生成模块化描述的网络，第二个参数指定了采样时间，它通常情况下为一正数。

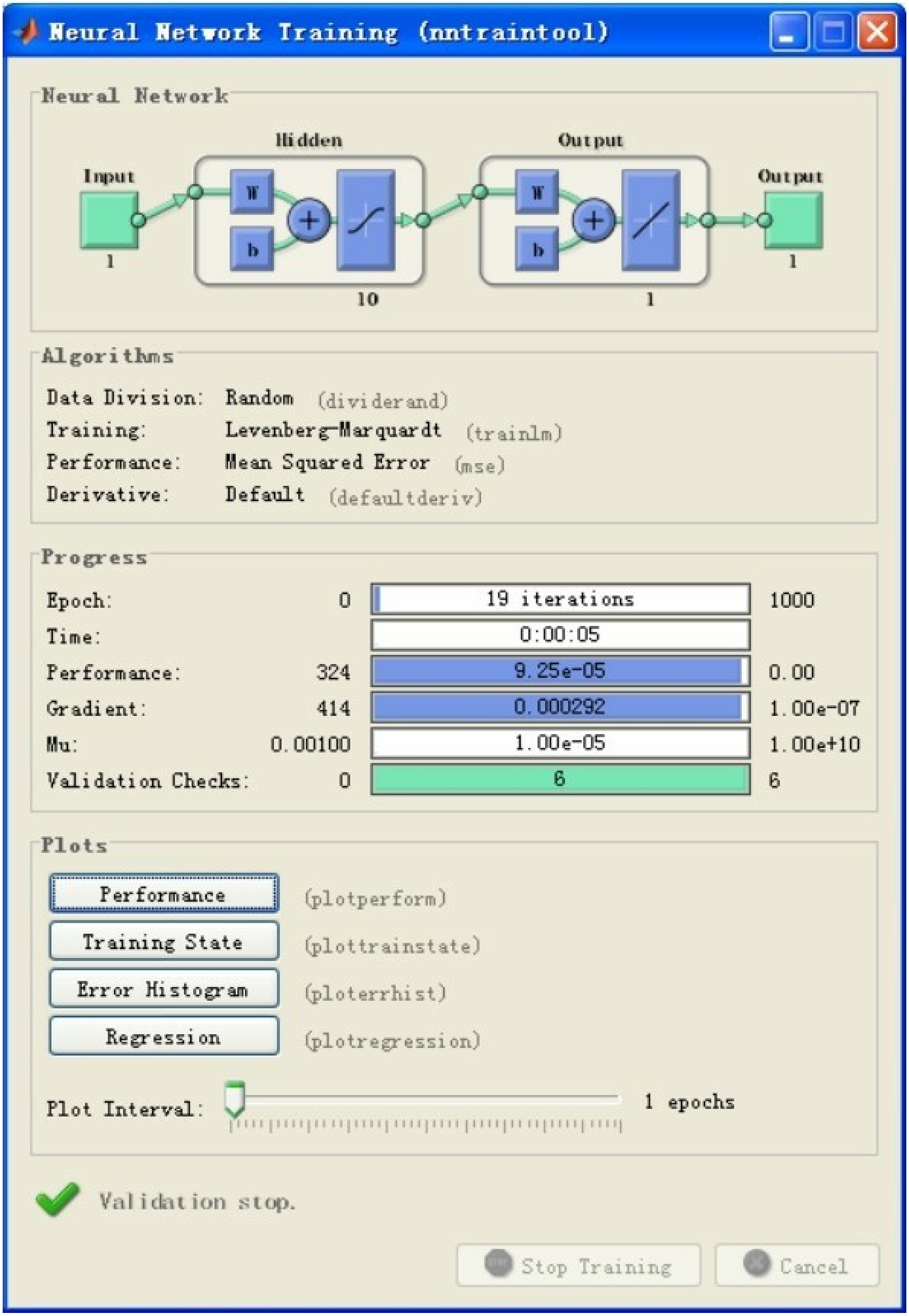
说明：如果网络没有与输入权值或者层中权值相关的延迟，则指定第二个参数为−1，那么函数gensim( )将生成一个连续采样的网络。

【例11-2】设计一个神经网络，并生成其模块化描述。定义网络的输入为x=[1 2 3 4 5]，相应的目标为t=[1 3 5 7 9]。

解：实现以上任务的MATLAB命令如下。

```
clear all
clc
x=[1 2 3 4 5]
t=[1 3 5 7 9]
net = feedforwardnet(10);
net = train(net,x,t)
gensim(net)
```

神经网络训练窗口如图11-19所示。



Neural Network Training (nntraintool)
Neural Network
Input
Hidden
Output
Output
W
b
W
b
1
10
1
1
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)
Progress
Epoch: 0 19 iterations 1000
Time: 0:00:05
Performance: 324 9.25e-05 0.00
Gradient: 414 0.000292 1.00e-07
Mu: 0.00100 1.00e-05 1.00e+10
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Plot Interval: 1 epochs
Validation stop.
Stop Training
Cancel

图11-19 神经网络训练窗口

在MATLAB中输入net，可以得到生成的神经网络信息，如下所示。

```
>> net
net =
    Neural Network
        name: 'Feed-Forward Neural Network'
        efficiency: .cacheDelayedInputs, .flattenTime,
        .memoryReduction, .flattenedTime
        userdata: (your custom info)
    dimensions:
        numInputs: 1
        numLayers: 2
        numOutputs: 1
        numInputDelays: 0
        numLayerDelays: 0
        numFeedbackDelays: 0
numWeightElements: 31
    sampleTime: 1
    connections:
        biasConnect: [1; 1]
        inputConnect: [1; 0]
        layerConnect: [0 0; 1 0]
        outputConnect: [0 1]
    subobjects:
        inputs:{1x1 cell array of 1 input}
        layers:{2x1 cell array of 2 layers}
```

outputs:{1x2 cell array of 1 output}

biases:{2x1 cell array of 2 biases}

inputWeights:{2x1 cell array of 1 weight}

layerWeights:{2x2 cell array of 1 weight}

functions:

adaptFcn: 'adaptwb'

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: 'dividerand'

divideParam: .trainRatio, .valRatio, .testRatio

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mse'

performParam: .regularization, .normalization

plotFcns:{'plotperform', plottrainstate, ploterrhist,plotregression}

plotParams:{1x4 cell array of 4 params}

trainFcn: 'trainlm'

trainParam: .showWindow, .showCommandLine, .show, .epochs, .time, .goal, .min_grad, .max_fail, .mu, .mu_dec, .mu_inc, .mu_max

weight and bias values:

IW:{2x1 cell} containing 1 input weight matrix

LW:{2x2 cell} containing 1 layer weight matrix

b:{2x1 cell} containing 2 bias vectors

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate: outputs=net(inputs)

得到模型中使用的模块库如图11-20所示。

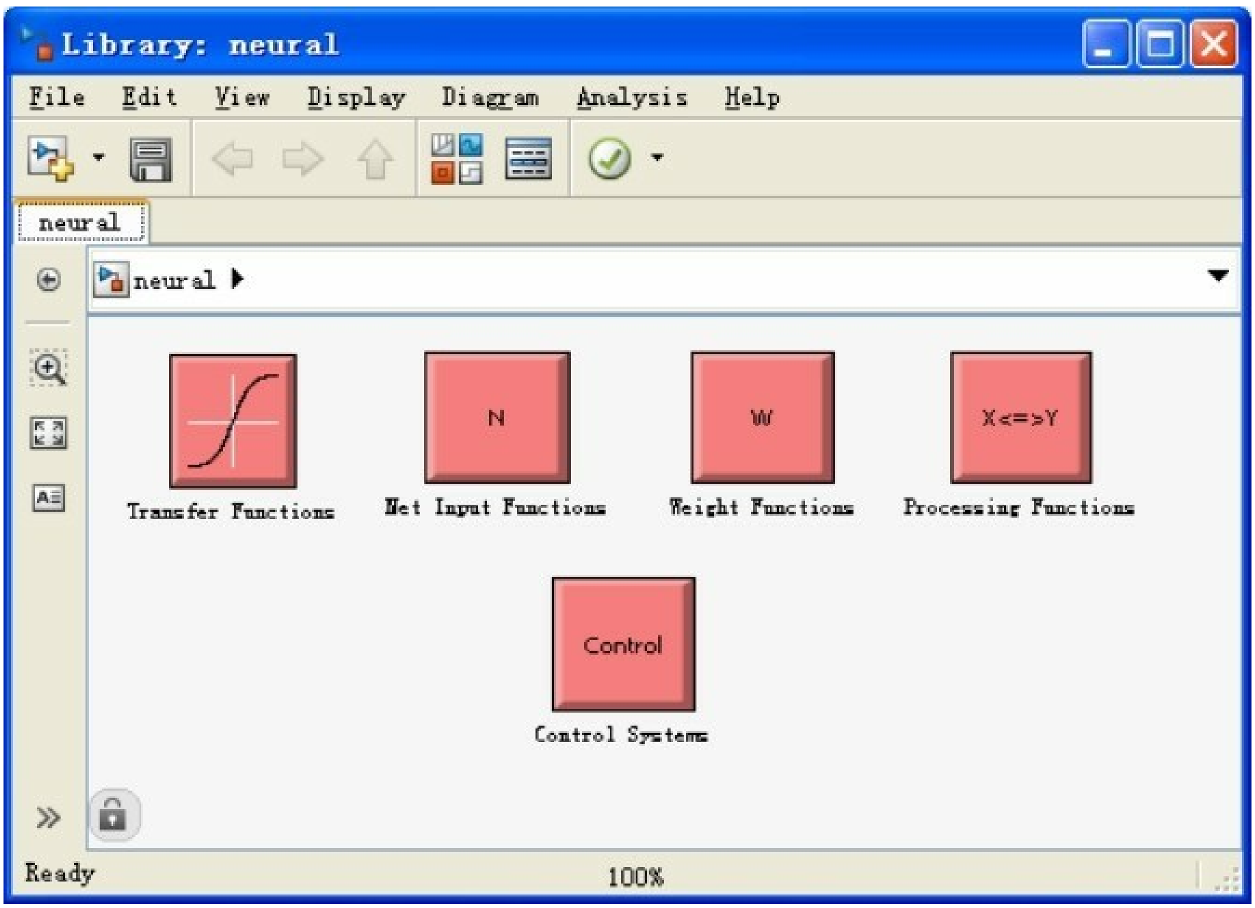


图11-20 模型中使用的模块库

得到建立的模型如图11-21所示。

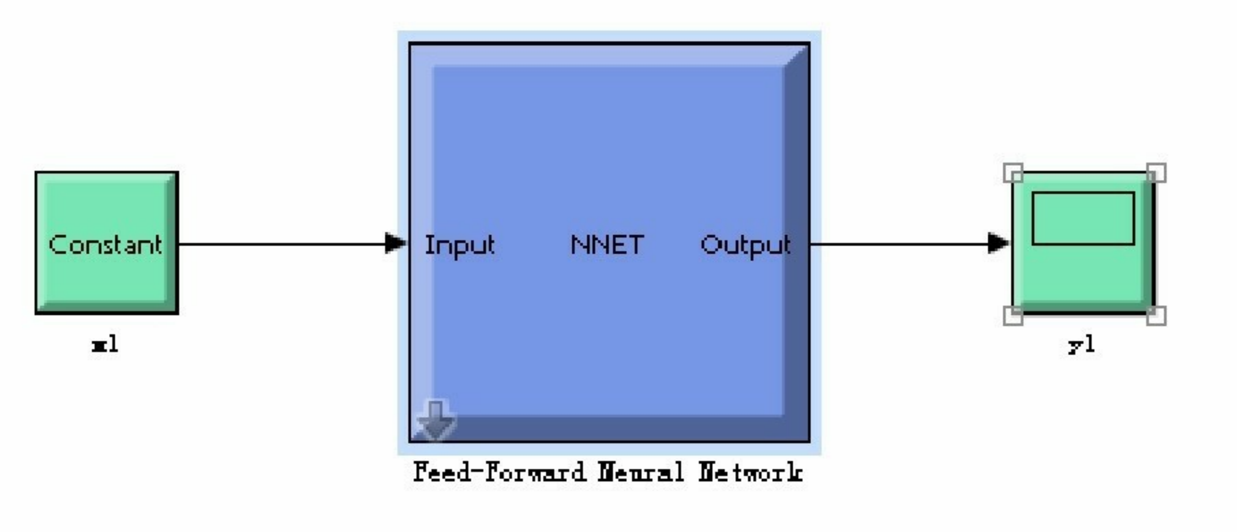


图11-21 建立的模型

通过sim函数，可以得到网络仿真结果，具体代码如下所示。

>> y=sim(net,x)

得到仿真结果如下所示。

y =

1.6961   2.4839   4.9928   7.0105   8.8899

题中的目标函数为t=[1 3 5 7 9]，比较y 和t 的值，前两个值的绝对误差均在0.5 左右，但是后面3个值的绝对误差均在0.2以内，说明建立的神经网络基本满足需要。

为了确定建立的Simulink模型是否满足目标，建立图11-22所示模型。

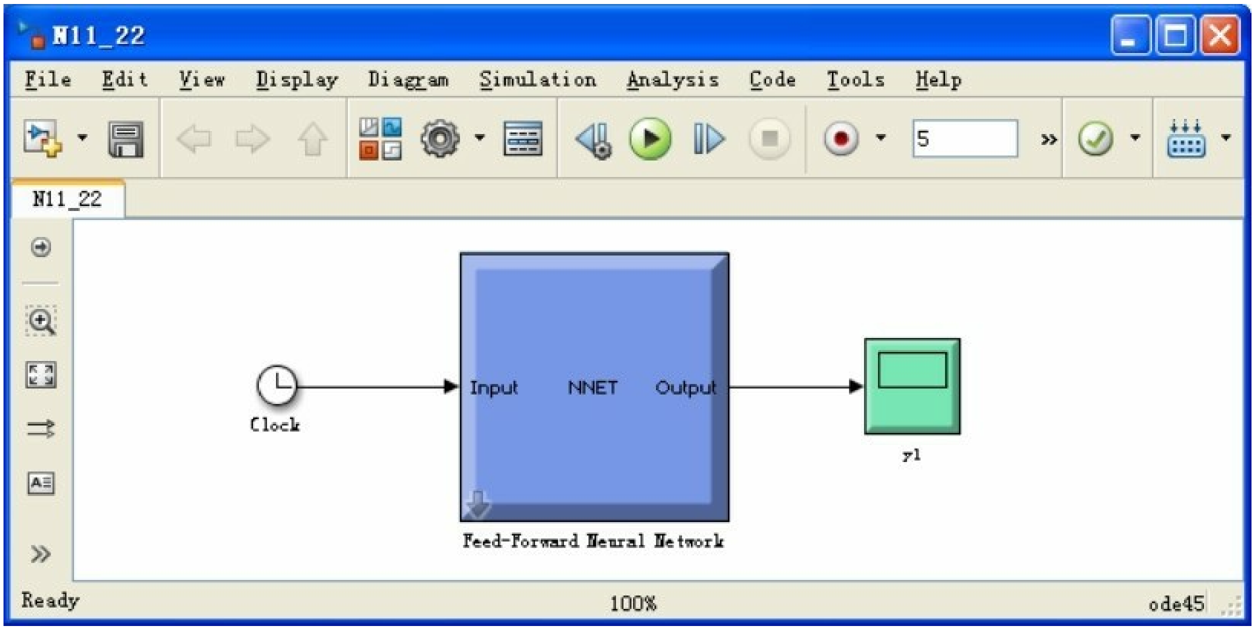


图11-22 验证模型

得到输出变量y1的曲线如图11-23所示。

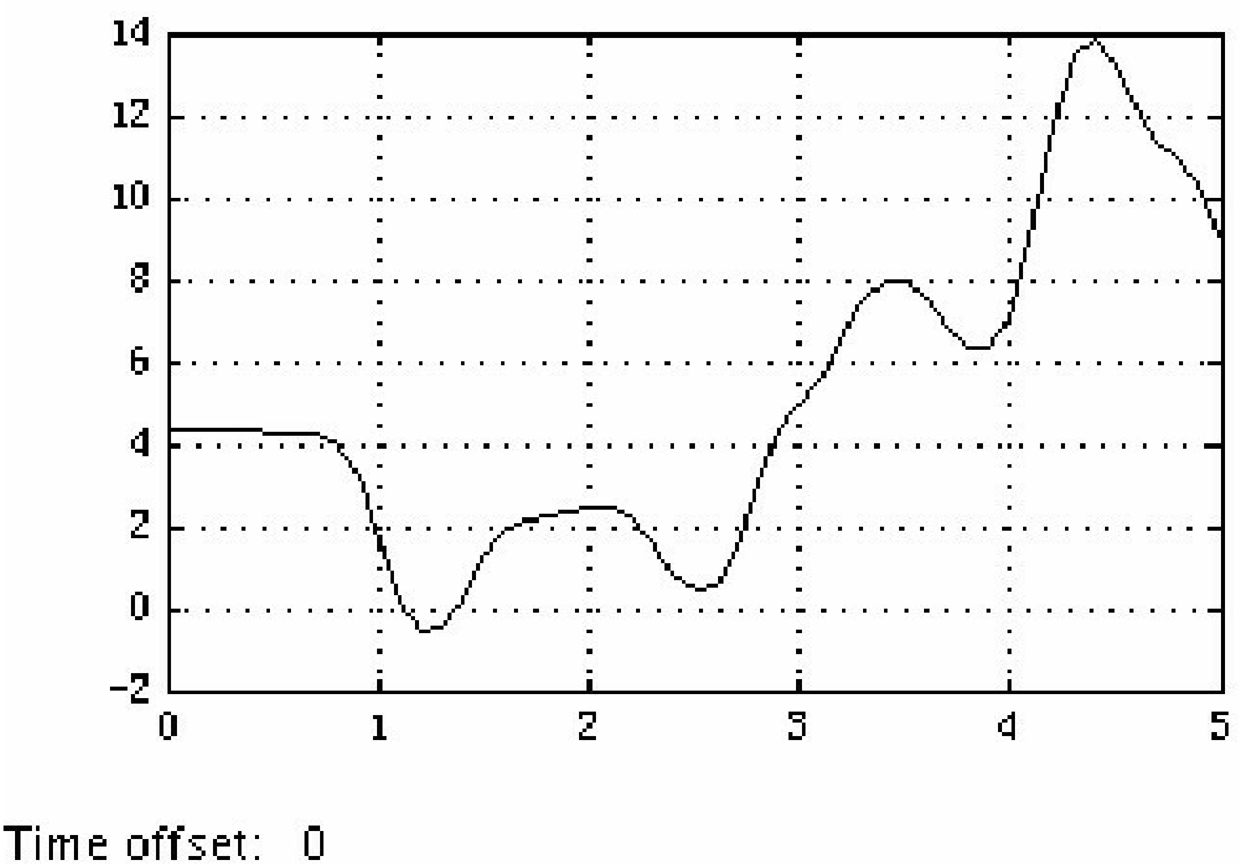


图11-23 输出变量y1的曲线

由图11-23可以看出，输出变量y1的值与用函数sim仿真得到的结果基本一致，说明题中建立的模型是正确的，且输出结果满足目标。

11.2 基于Simulink的神经网络控制系统

神经网络在系统辨识和动态系统控制中已经得到了非常成功的使用。由于神经网络具有全局逼近能力，使得其在对非线性系统建模和对一般情况下的非线性控制器的实现等方面而应用得比较普遍。

本节将介绍3 种在神经网络工具箱的控制系统模块（Control Systems）中利用Simulink实现的比较普遍的神经网络结构，它们常用于预测和控制，并已在MATLAB对应的神经网络工具箱中给出了实现。

这3种神经网络结构如下。

·神经网络模型预测控制（NN Predictive Controller）

·反馈线性化控制（NARMA-L2 Controller）

·模型参考控制（Model Reference Controller）

使用神经网络进行控制时，通常有两个步骤：系统辨识和控制设计。

在系统辨识阶段，主要任务是对需要控制的系统建立神经网络模型；在控制设计阶段，主要使用神经网络模型来设计（训练）控制器。

说明：在本节将要介绍的3种控制网络结构中，系统辨识阶段是相同的，而控制设计阶段则各不相同。

对于模型预测控制，系统模型用于预测系统未来的行为，并且找到最优的算法，用于选择控制输入，以优化未来的性能。

对于NARMA-L2（反馈线性化）控制，控制器仅仅是将系统模型进行重整。

对于模型参考控制，控制器是一个神经网络，它被训练以用于控制系统，使得系统跟踪一个参考模型，这个神经网络系统模型在控制器训练中起辅助作用。

11.2.1 神经网络模型预测控制

神经网络预测控制器是使用非线性神经网络模型来预测未来模型性能。控制器计算控制输入，而控制输入在未来一段指定的时间内将最优化模型性能。模型预测第一步是要建立神经网络模型（系统辨识）；第二步，使用控制器来预测未来神经网络性能。

1.系统辨识

模型预测的第一步就是训练神经网络未来表示网络的动态机制。模型输出与神经网络输出之间的预测误差，用来作为神经网络的训练信号，该过程用图11-24来表示。

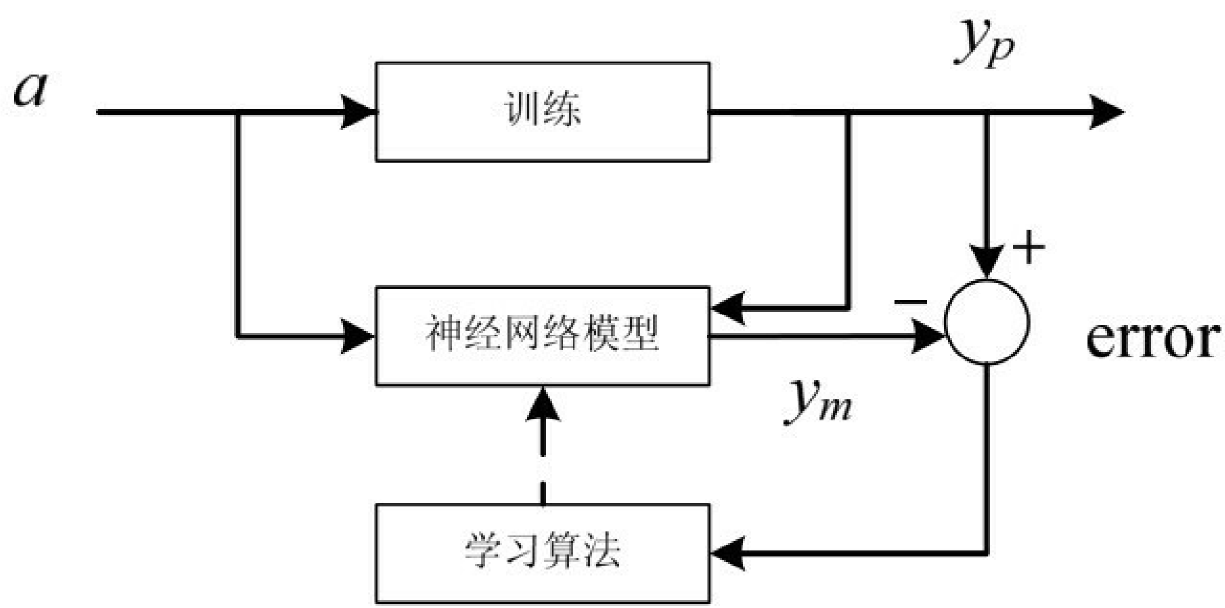


图11-24 训练神经网络

神经网络模型利用当前输入和当前输出预测神经未来输出值。神经网络模型结构如图11-25所示，该网络可以批量在线训练。

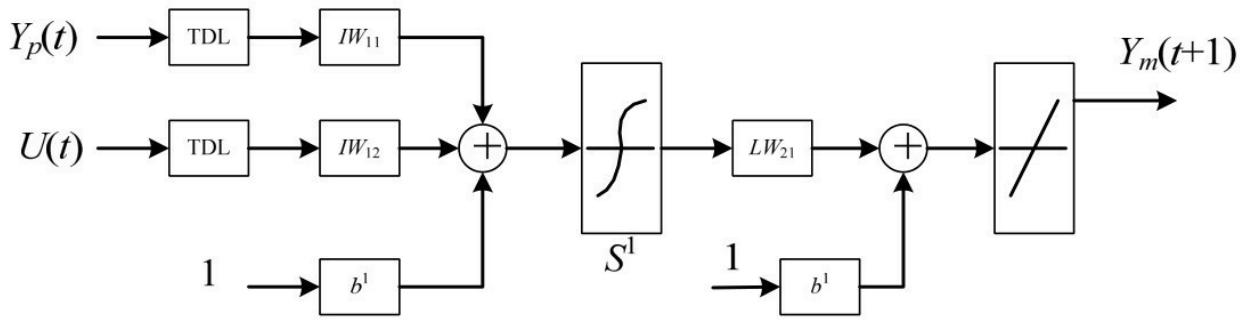


图11-25 神经网络模型结构

2.模型预测

模型预测方法是基于水平后退的方法，神经网络模型预测在指定时间内预测模型响应。预测使用数字最优化程序来确定控制信号，最优化的性能准则函数如下。

$$J=\sum_{j=1}^{N_2}[y_r(t+j)-y_m(t+j)]^2+\rho\sum_{j=1}^{N_u}[u(t+j-1)-u(t+j-2)]^2$$

式中 N_2为预测时域长度；N_u为控制时域长度；u(t) 为控制信号；y_r为期望响应，y_m为网络模型响应，ρ为控制量加权系数。

图 11-26 描述了模型预测控制的过程。控制器由神经网络模型和最优化方块组成，最优化方块确定 u（通过最小化 J），最优 u 值作为神经网络模型的输入，控制器方块可用Simulink实现。

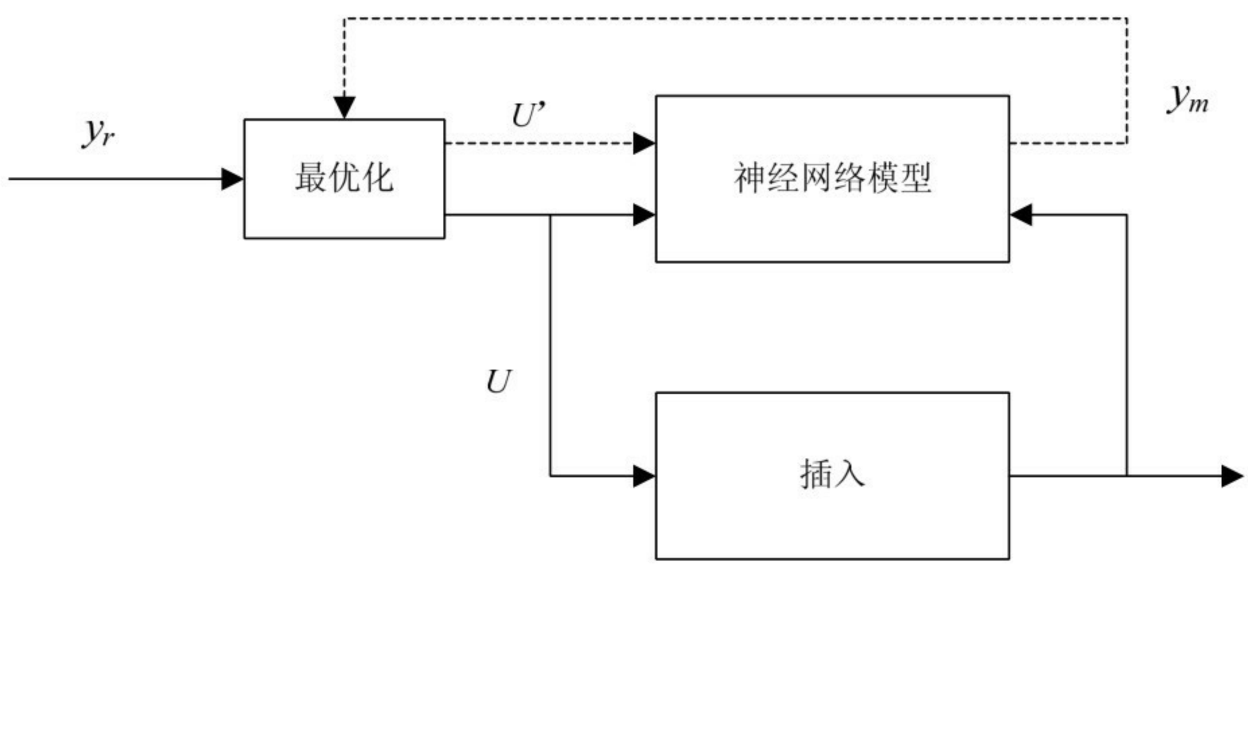


图11-26 预测模型控制的过程

在 MATLAB 神经网络工具箱中实现的神经网络预测控制器使用了一个非线性系统模型，用于预测系统未来的性能。

接下来这个控制器将计算控制输入，用于在某个未来的时间区间里优化系统的性能。进行模型预测控制首先要建立系统的模型，然后使用控制器来预测未来的性能。

下面结合MATLAB神经网络工具箱中提供的一个演示实例，介绍Simulink中的实现过程。

（1）问题的描述。

要讨论的问题基于一个搅拌器（CSTR），如图11-27所示。

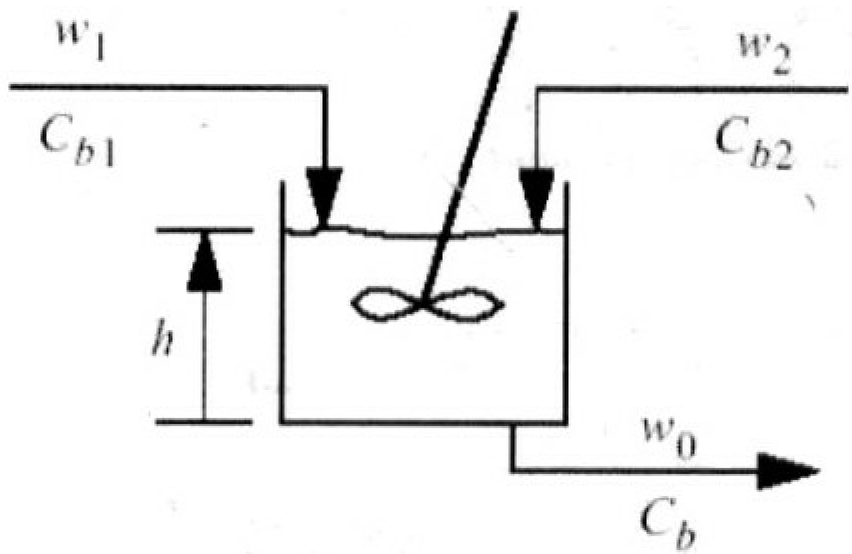


图11-27 搅拌器

对于这个系统，其动力学模型为

$$\frac{dh(t)}{dt}=w_1(t)+w_2(t)-0.2\sqrt{h(t)}$$

$$\frac{dC_b(t)}{dt}=(C_{b1}-C_b(t))\frac{w_1(t)}{h(t)}+(C_{b2}-C_b(t))\frac{w_2(t)}{h(t)}-\frac{k_1C_b(t)}{(1+k_2C_b(t))^2}$$

其中h(t)为液面高度，$C_b(t)$为产品输出浓度，$w_1(t)$为浓缩液C_{b1}的输

入流速，$w_2(t)$为稀释液C_{b2}的输入流速。输入浓度设定为：$C_{b1}=24.9$，$C_{b2}=0.1$。消耗常量设置为：$k_1=1$，$k_2=1$。

控制的目标是通过调节流速 $w_2(t)$来保持产品浓度。为了简化演示过程，不妨设$w_1(t)=0.1$。在本例中不考虑液面高度h(t)。

（2）建立模型。

在 MATLAB 神经网络工具箱中提供了这个演示实例。只需在MATLAB命令窗口中输入命令predcstr，就会自动地调用Simulink，并且产生如图11-28所示的模型窗口。

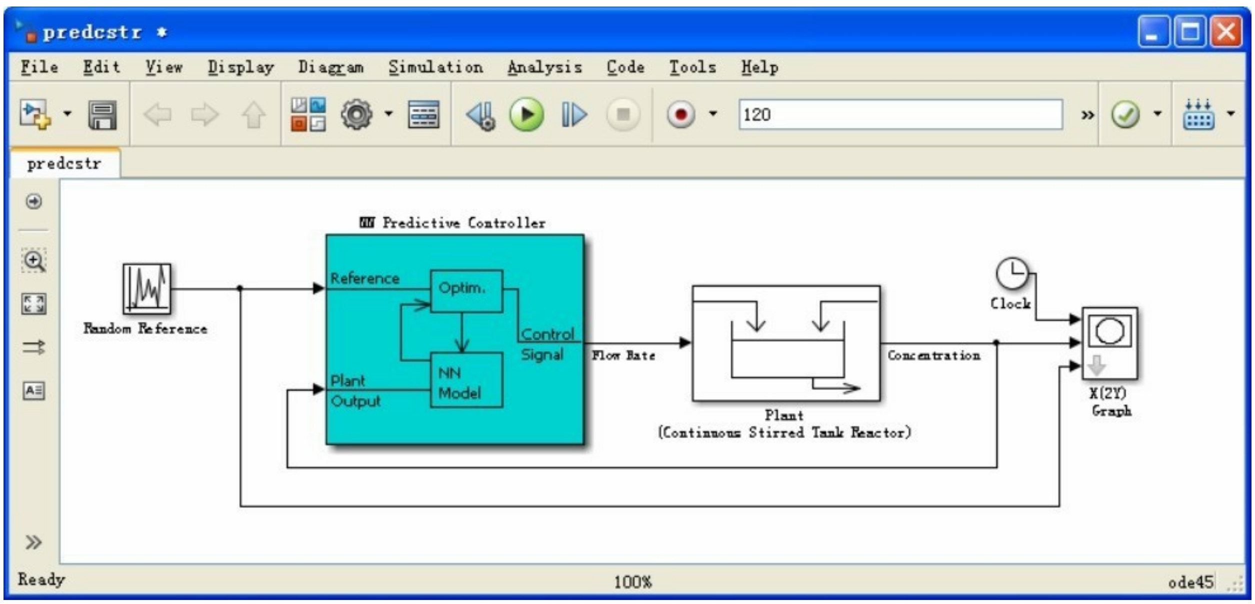


图11-28 模型窗口

其中神经网络预测控制模块（NN Predctive Controller）和X(2Y) Graph 模块由神经网络模块集（Neural Network Blockset）中的控制系统模块库（Control Systems）复制而来。

图11-28中的Plant（Continuous Stirred Tank Reactor）模块包含了搅拌器系统的Simulink模型。双击这个模块，可以得到具体的Simulink实现，此处不加以深入讨论。

NN Predictive Controller模块的Control Signal 端连接到搅拌器系统模型的输入端，同时搅拌器系统模型的输出端连接到NN Predictive

Controller 模块的Plant Output 端，参考信号连接到NN Predictive Controller模块的Reference端。

双击 NN Predctive Controller 模块，将会产生一个神经网络预测控制器参数设置窗口（Neural Network Predctive Control），如图11-29 所示。这个窗口用于设计模型预测控制器。

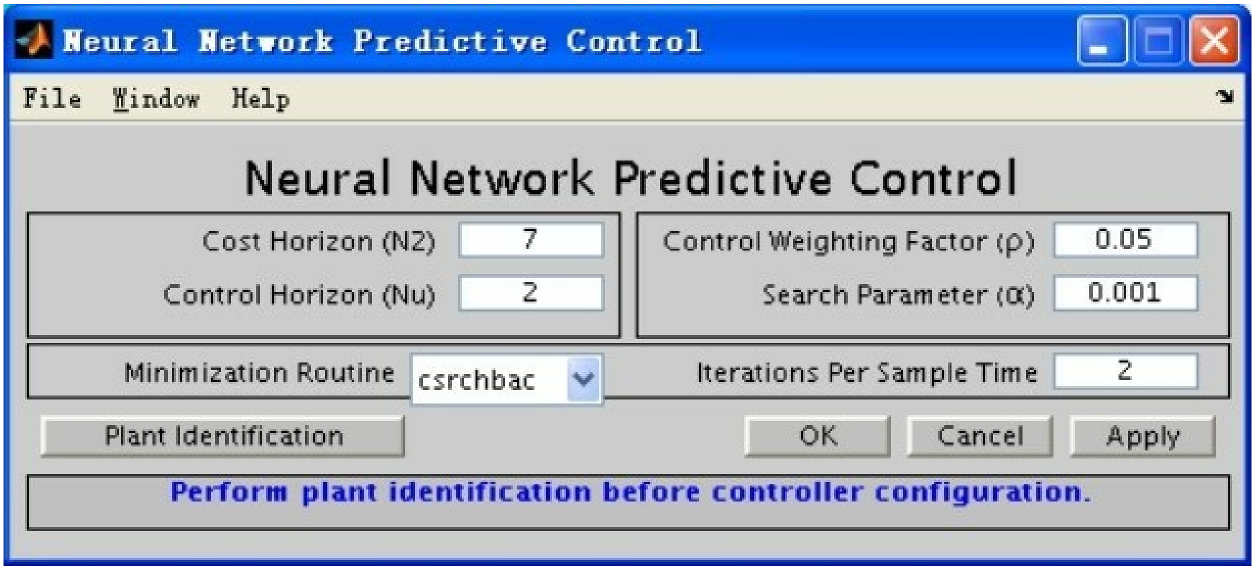


图11-29 神经网络模型预测控制器参数设置窗口

在这个窗口中，有多项参数可以调整，用于改变预测控制算法中的有关参数。将鼠标移到相应的位置，就会出现对这一参数的说明。

（3）系统辨识。

在神经网络预测控制器的窗口中单击“Plant Identification”按钮，将产生一个模型辨识参数设置窗口（Plant Identification），用于设置系统辨识的参数，如图11-30 所示。

（4）系统仿真。

在Simulink模型窗口中，选择Simulation→parameter命令设置相应的仿真参数，然后单击开始命令进行仿真。仿真的过程需要一段时间。当仿真结束时，将会显示出系统的输出和参考信号，如图11-31所示。

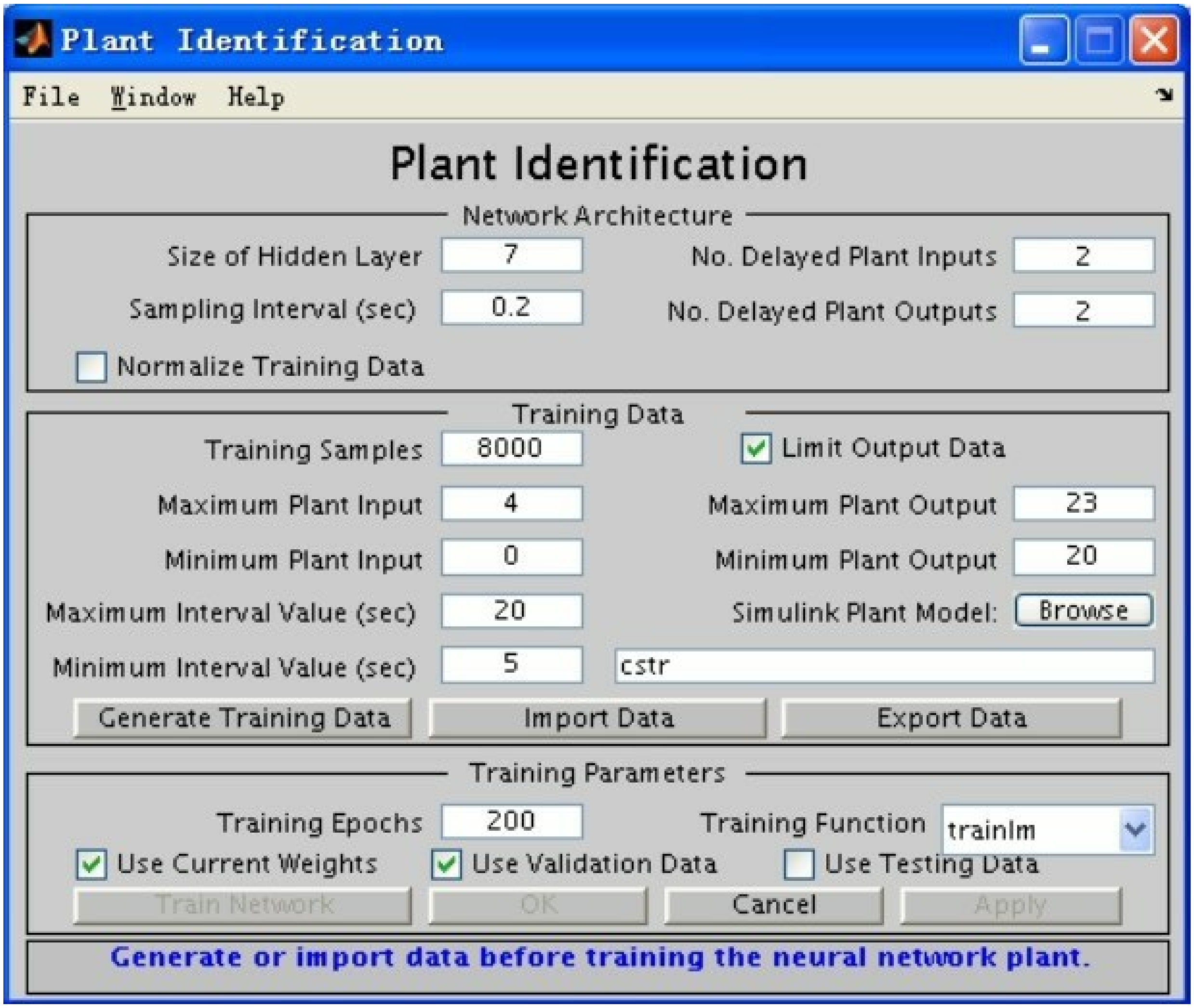


图11-30 模型辨识参数设置窗口

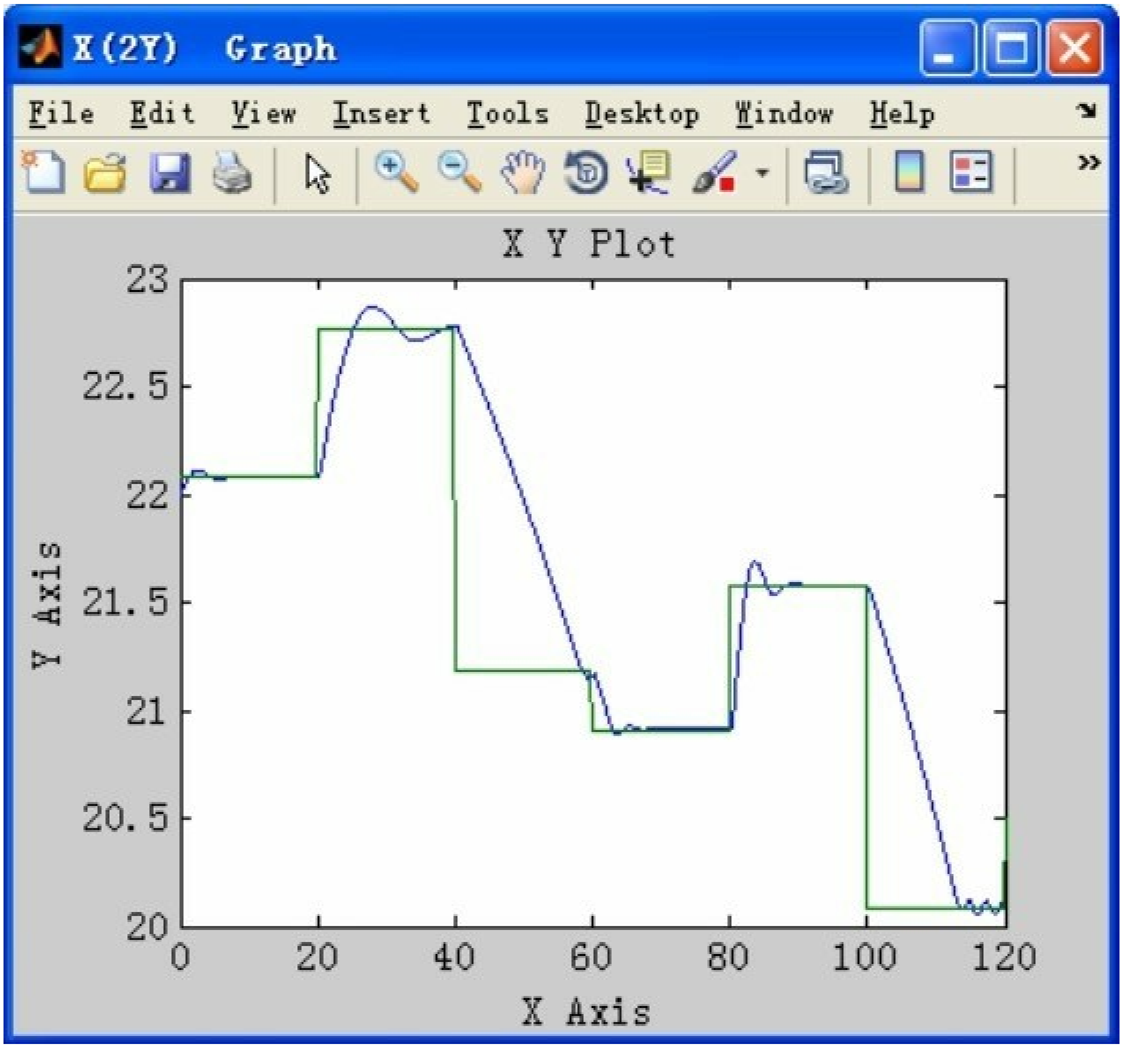


图11-31 输出和参考信号

（5）数据保存。

在图11-31 中，利用File选项下的“Save Workspace As...”命令，可以将设计好的网络训练数据保存到工作空间中或是保存到磁盘文件中。

神经网络预测控制是使用神经网络系统模型来预测系统未来的行为。优化算法用于确定控制输入，这个控制输入优化了系统在一个有限时间段里的性能。系统训练仅仅需要对于静态网络的成批训练算法，当然，训练速度非常快。控制器不要在线的优化算法，这就需要比其用户控制器更多的计算。

11.2.2 反馈线性化控制

反馈线性化（NARMA-L2）的中心思想是通过去掉非线性，将一个非线性系统变换成线性系统。

辨识 NARMA-L2 模型与模型预测控制一样，反馈线性化控制的第一步就是辨识被控制的系统。通过训练一个神经网络来表示系统的前向动态机制，在第一步中首先选择一个模型结构以供使用。一个用来代表一般的离散非线性系统的标准模型是非线性自回归移动平均模型（NARMA），用下式来表示。

$$y(k+d)=N[y(k),y(k-1),\cdots,y(k-n+1),u(k),u(k-1),\cdots,u(k-n+1)]$$

式中，u(k)表示系统的输入，y(k)表示系统的输出。在辨识阶段，训练神经网络使其近似等于非线性函数N。

如果希望系统输出跟踪一些参考曲线$y(k+d)=y_r(k+d)$，下一步就是建立一个有如下形式的非线性控制器。

$$u(k)=G[y(k),y(k-1),\cdots,y(k-n+1),y_r(k+d),u(k-1),\cdots,u(k-n+1)]$$

使用该类控制器的问题是，如果想训练一个神经网络用来产生函数G（最小化均方差），必须使用动态反馈，该过程相当慢。Narendra 和 Mukhopadhyay 提出的一个解决办法是，使用近似模型来代表系统。

在这里使用的控制器模型是基于NARMA-L2近似模型

$$\begin{aligned}\hat{y}(k+d)=&f[y(k),y(k-1),\cdots,y(k-n+1),u(k-1),\cdots,u(k-n+1)]\\&+g[y(k),y(k-1),\cdots,y(k-n+1),u(k-1),\cdots,u(k-n+1)]u(k)\end{aligned}$$

该模型是并联形式，控制器输入 u(k)没有包含在非线性系统里。这种形式的优点是，能解决控制器输入使系统输出跟踪参考曲线$y(k+d)=y_r(k+d)$。

最终的控制器形式如下。

$$u(k)=\frac{y_r(k+d)-f[y(k),y(k-1),\cdots,y(k-n+1),u(k),u(k-1),\cdots,u(k-n+1)]}{g[y(k),y(k-1),\cdots,y(k-n+1),u(k),u(k-1),\cdots,u(k-n+1)]}$$

直接使用该等式会引起实现问题，因为基于输出 y(k)的同时必须得到 u(k)，所以采用下述模型。

$$\begin{aligned} y(k+d) &= f[y(k), y(k-1), \cdots, y(k-n+1), u(k), \cdots, u(k-n+1)] \\ &\quad + g[y(k), y(k-1), \cdots, y(k-n+1), u(k), \cdots, u(k-n+1)] \cdot u(k+1) \end{aligned}$$

式中d≥2。

利用NARMA-L2模型，可得到如下的NARMA-L2控制器。

$$u(k+1) = \frac{y_\tau(k+d) - f[y(k), y(k-1), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-n+1)]}{g[y(k), y(k-1), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-n+1)]}$$

式中d≥2。

下面利用NARMA-L2（反馈线性化）控制分析磁悬浮控制系统。

（1）问题的描述。

如图 11-32 所示，有一块磁铁，被约束在垂直方向上运动。在其下方有一块电磁铁，通电以后，电磁铁就会对其上的磁铁产生小电磁力作用。目标就是通过控制电磁铁，使得其上的磁铁保持悬浮在空中，不会掉下来。

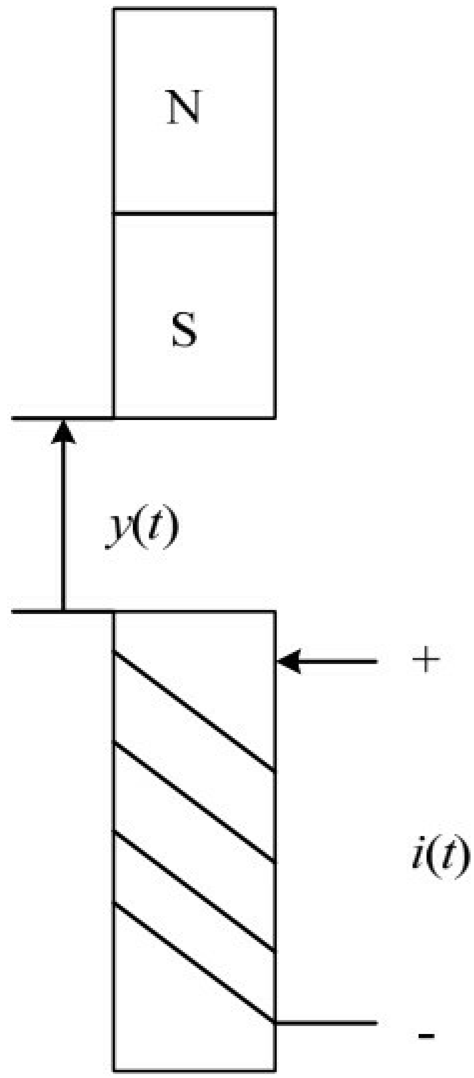


图11-32 悬浮磁铁控制系统

建立这个实际问题的动力学方程如下。

$$\frac{\mathrm{d}^2 y(t)}{\mathrm{d}t^2} = -g + \frac{\alpha i^2(t)}{My(t)} - \frac{\beta}{M}\frac{\mathrm{d}\,y(t)}{\mathrm{d}t}$$

式中 y(t)表示磁铁离电磁铁的距离，i(t)代表电磁铁中的电流，M 代表磁铁的质量，g代表重力加速度，β代表黏性摩擦系数，它由磁铁所在的容器的材料决定；α代表场强常数，它由电磁铁上所绕的线圈圈数，

以及磁铁的强度所决定。

（2）建立模型。

MATLAB的神经网络工具箱中提供了这个演示实例。只需在MATLAB命令窗口中输入narmamaglev，就会自动地调用Simulink，并且产生如图11-33所示模型窗口。

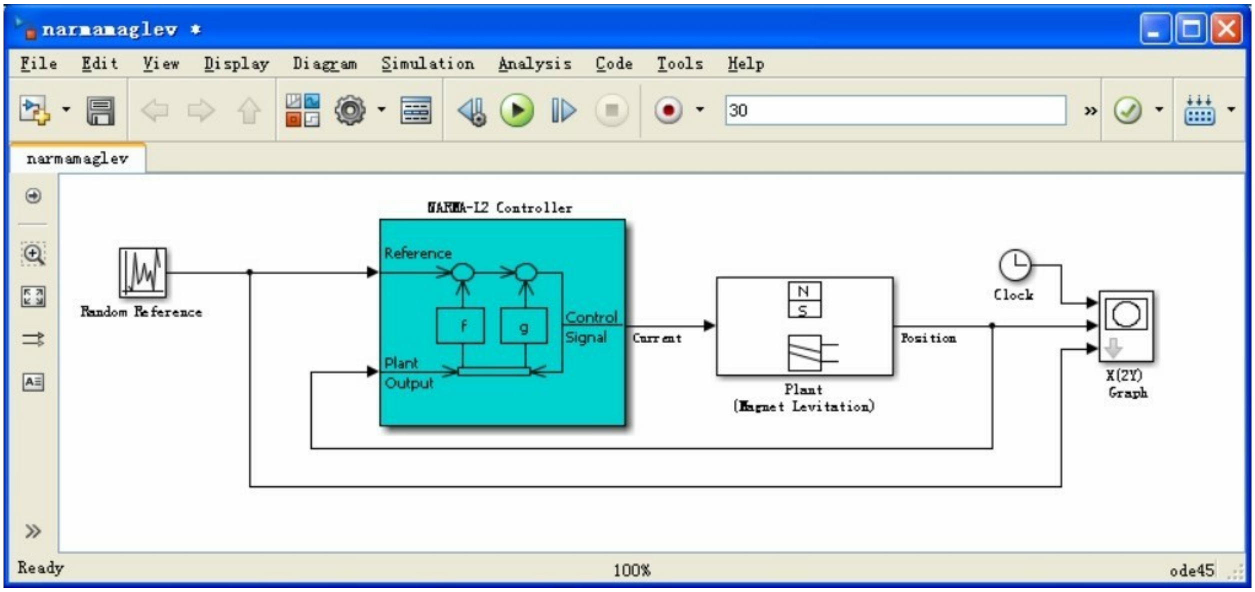


图11-33 模型窗口

（3）系统辨识。

双击NARMA-L2 Controller模块，将会产生一个新的窗口，如图11-34 所示。

（4）系统仿真。

在Simulink模型窗口中，选择Simulation→parameter命令设置相应的仿真参数，然后单击Start命令开始仿真。仿真的过程需要一段时间。当仿真结束时，将会显示出系统的输出和参考信号，如图11-35所示。

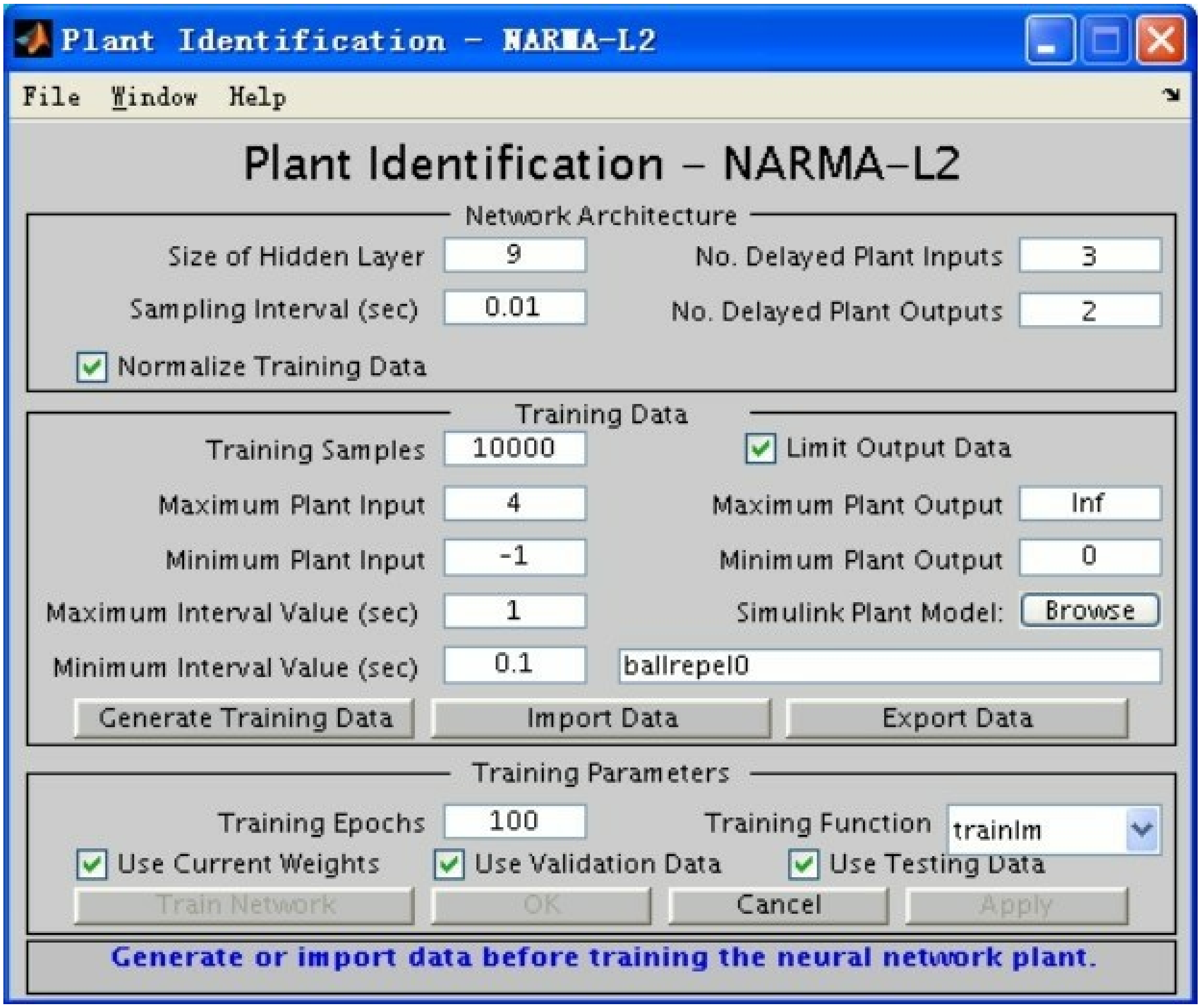


图11-34 系统辨识参数设置窗口

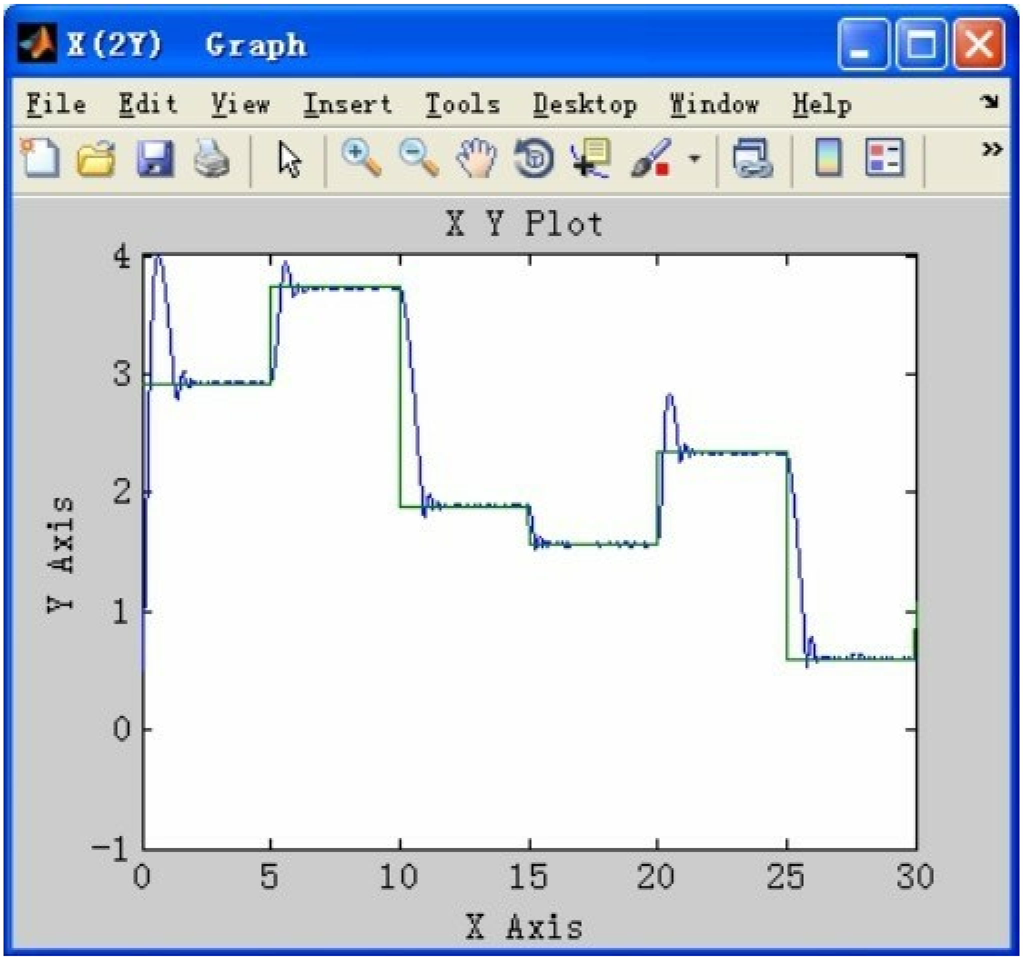


图11-35 输出和参考信号

11.2.3 模型参考控制

神经模型参考控制采用两个神经网络：一个控制器网络和一个实验模型网络，如图11-36所示。首先辨识出实验模型，然后训练控制器，使得实验输出跟随参考模型输出。

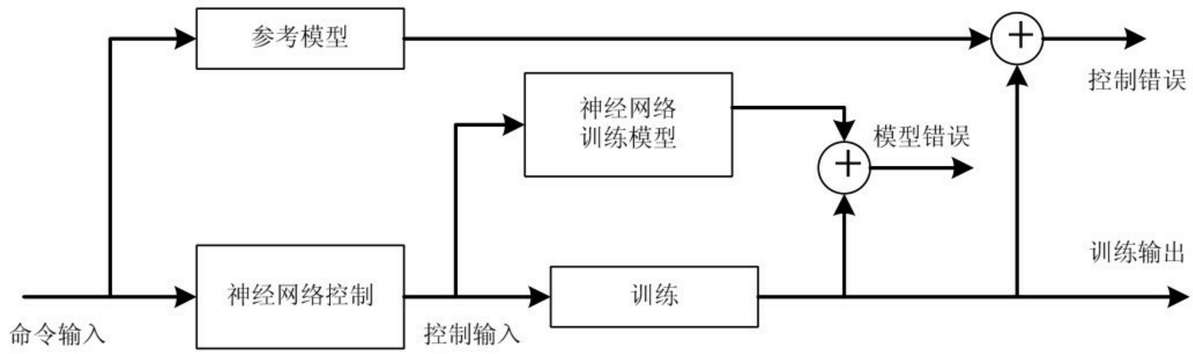


图11-36 神经模型参考控制系统

图 11-37 显示了神经网络实验模型的详细情况，每个网络由两层组成，并且可以选择隐含层的神经元数目。

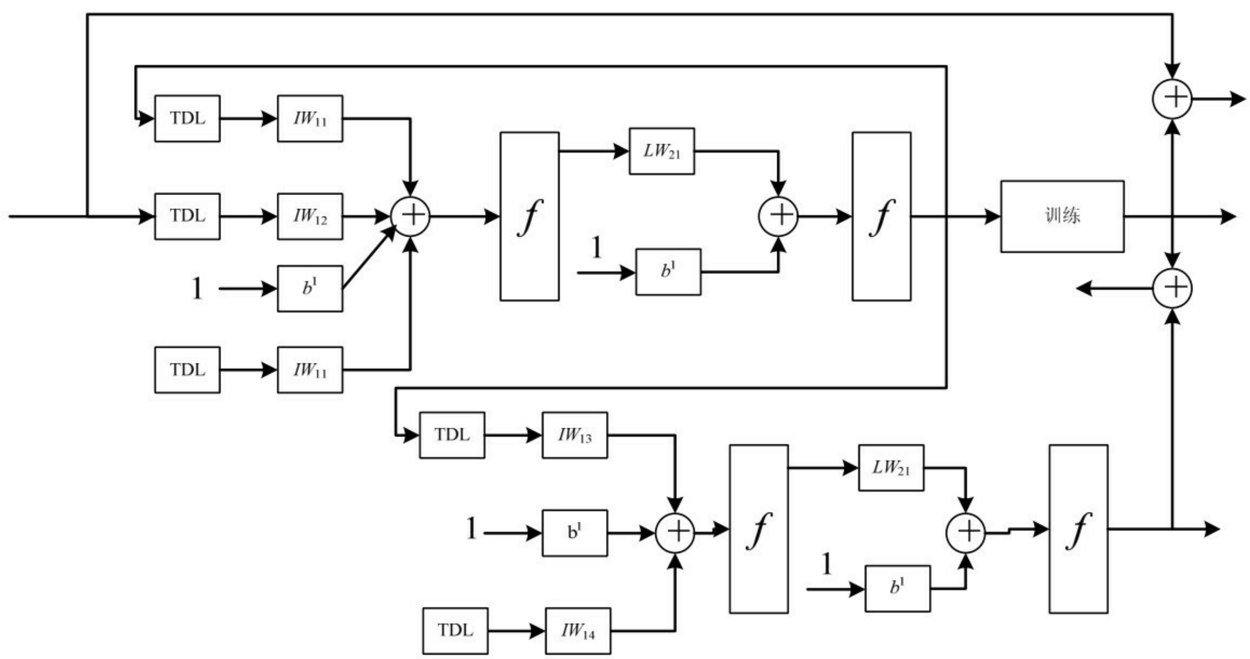


图11-37神经网络实验模型

有3组控制器输入：延迟的参考输入、延迟的控制输出和延迟的系统输出。对于每一个这种输入，可以选择延迟值。通常，随着系统阶次的增加，延迟的数目也增加。对于神经网络系统模型，有两组输入，即延迟的控制器输出和延迟的系统输出。

下面结合 MATLAB 神经网络工具箱中提供的一个实例，来介绍神经网络控制器的训练过程。

（1）问题的描述。

图11-38中显示了一个简单的单连接机械臂，目的是控制它的运动。

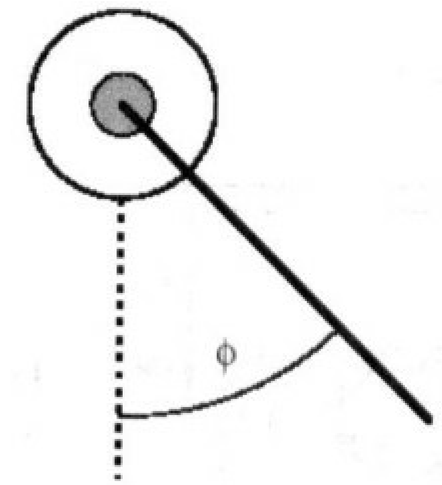


图11-38 简单的单连接机械臂

建立它的运动方程式，如下所示。

$$\frac{d^2\Phi}{dt^2} = -10\sin\Phi - 2\frac{d\Phi}{dt} + u$$

式中φ代表机械臂的角度，u代表DC（直流）电机的转矩。目标是训练控制器，使得机械臂能够跟踪参考模型

$$\frac{d^2 y_r}{dt^2} = -9y_r - 6\frac{dy_r}{dt} + 9r$$

式中 y_r代表参考模型的输出，r 代表参考信号。

（2）建立模型。

MATLAB的神经网络工具箱中提供了这个演示实例。控制器的输入包含了两个延迟参考输入、两个延迟系统输出和一个延迟控制器输出，采样间隔为0.05秒。

只需在MATLAT命令行窗口中输入mrefrobotarm，就会自动地调用

Simulink，并且产生如图11-39所示的模型窗口。

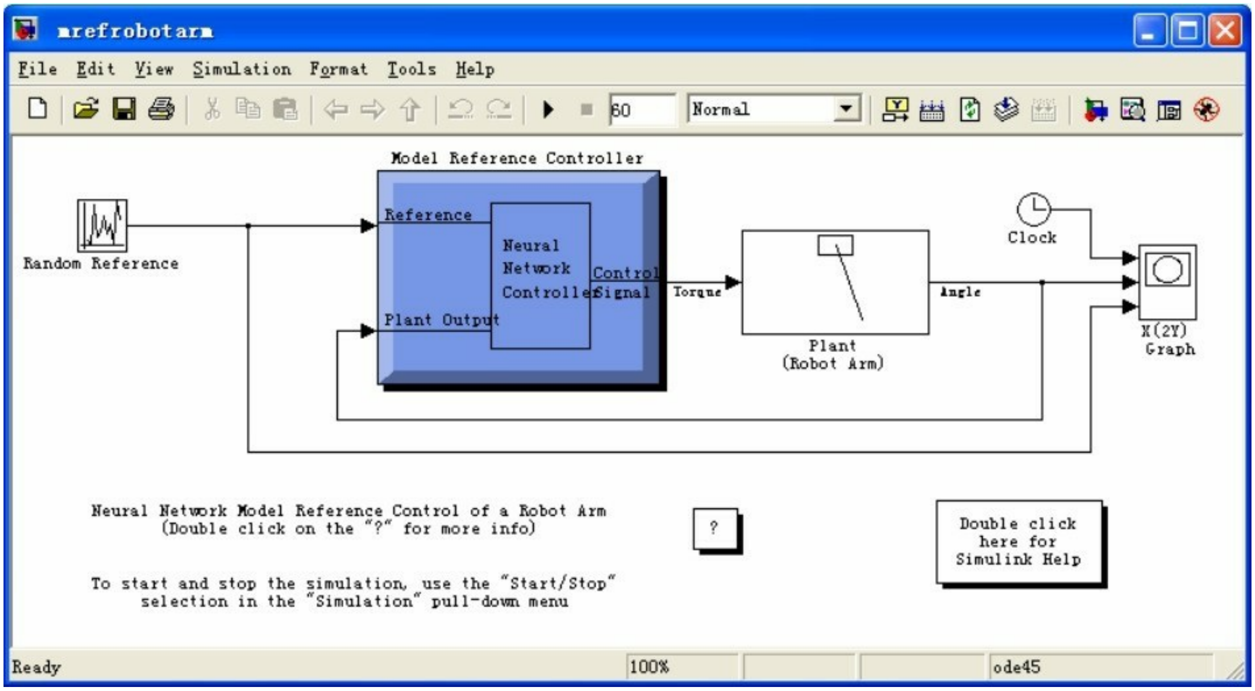


图11-39 模型窗口

双击模型参考控制模块，将会产生一个模型参考控制参数（Mode Reference Control）设置窗口，如图11-40所示。这个窗口用于训练模型参考神经网络。

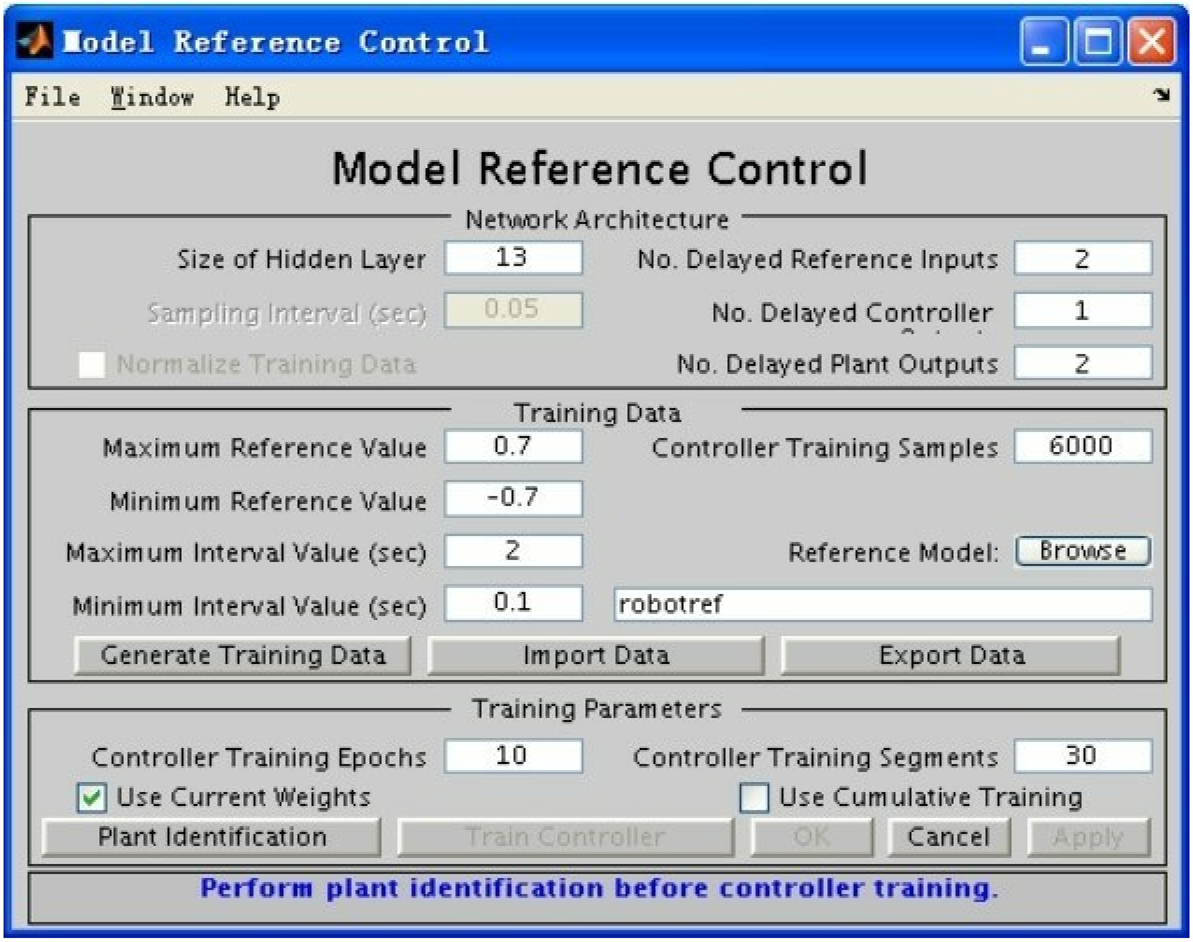


图11-40 模型参考控制参数设置窗口

在模型参考控制参数设置窗口中单击“Plant　Identification”按钮，将会弹出一个系统辨识窗口，如图11-41所示。当系统辨识结束后，单击图11-41中“OK”按钮，返回到模型参考控制窗口。

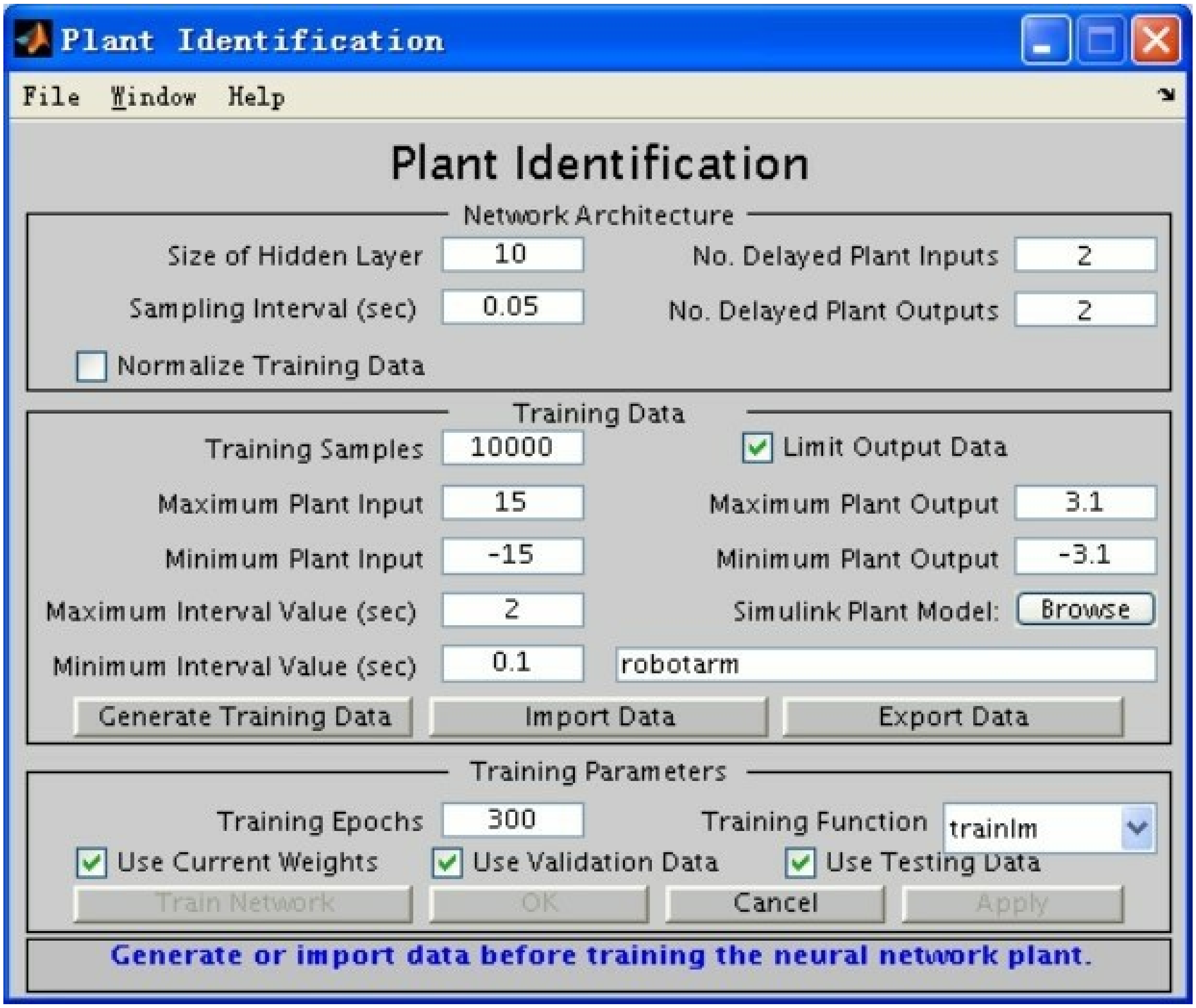


图11-41 系统辨识参数设置窗口

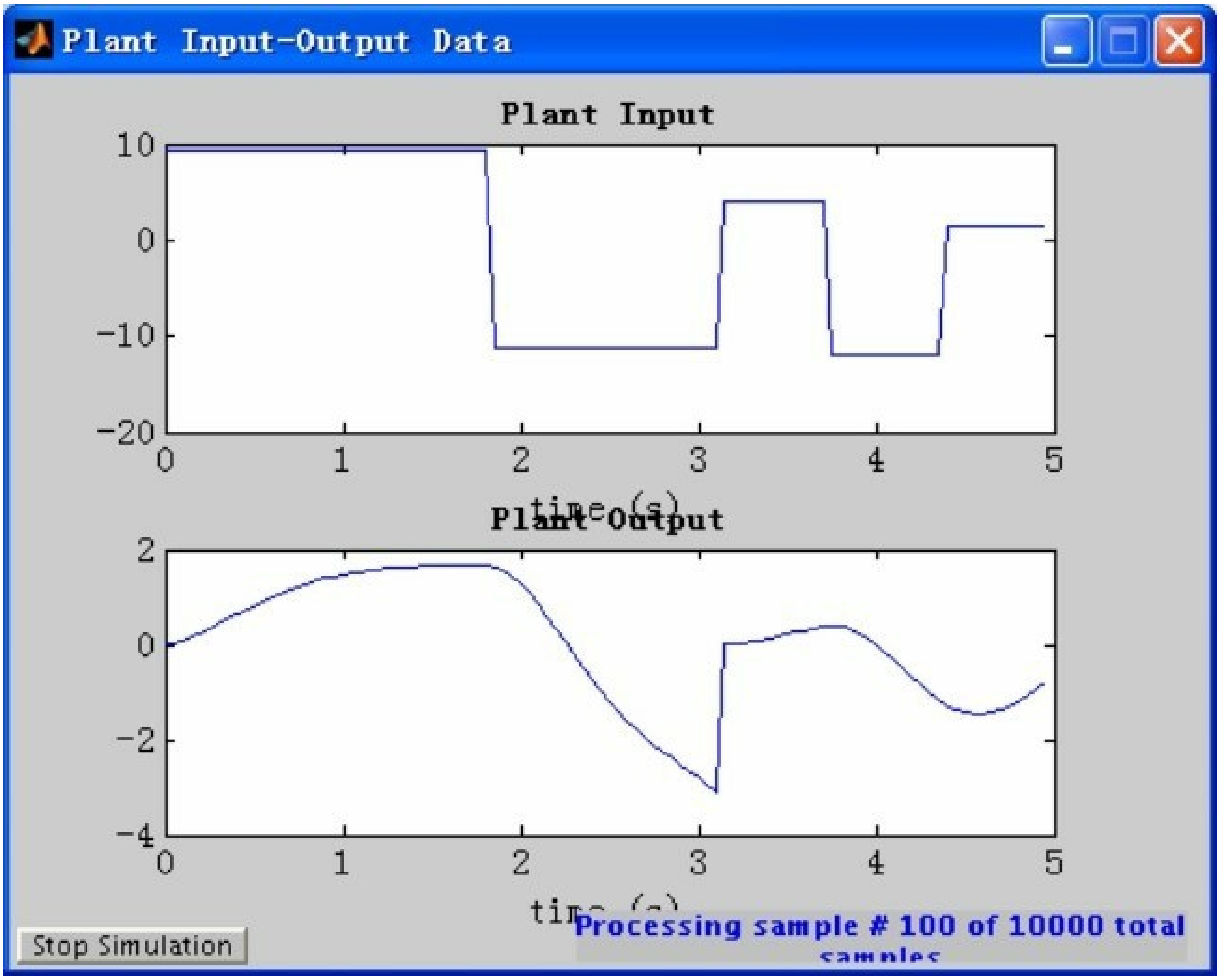


图11-42 训练过程误差曲线

当系统模型神经网络辨识完成后，在图 11-40 所示的模型参考控制窗口中单击“Generate Training Data”按钮，程序就会提供一系列随机阶跃信号，来对控制器产生训练数据，训练过程误差曲线如图11-42所示。

当接受这些数据后，就可以利用模型参考控制参数设置窗口中的“Train Controller”按钮对控制器进行训练。控制器训练需要的时间比系统模型训练需要的时间多得多。这是因为控制器必须使用动态反馈算法。

如果需要使用新的数据继续训练，可以在单击“Train Controller”按钮之前再次单击“Generate Training Data”按钮或者“Import Data”按钮（注

意：要确认Use Current Weights 被选中）。另外，如果系统模型不够准确，也会影响控制器的训练。

在模型参考控制窗口中单击“OK”按钮，将训练好的神经网络控制器权值导入 Simulink 模型窗口，并返回到Simulink模型窗口。

（3）系统仿真。

在 Simulink 模型窗口中，首先选择 Simulation→parameter命令设置相应的仿真参数，然后选择Start命令开始仿真。仿真的过程需要一段时间。当仿真结束时，将会显示出系统的输出和参考信号，如图11-43所示。

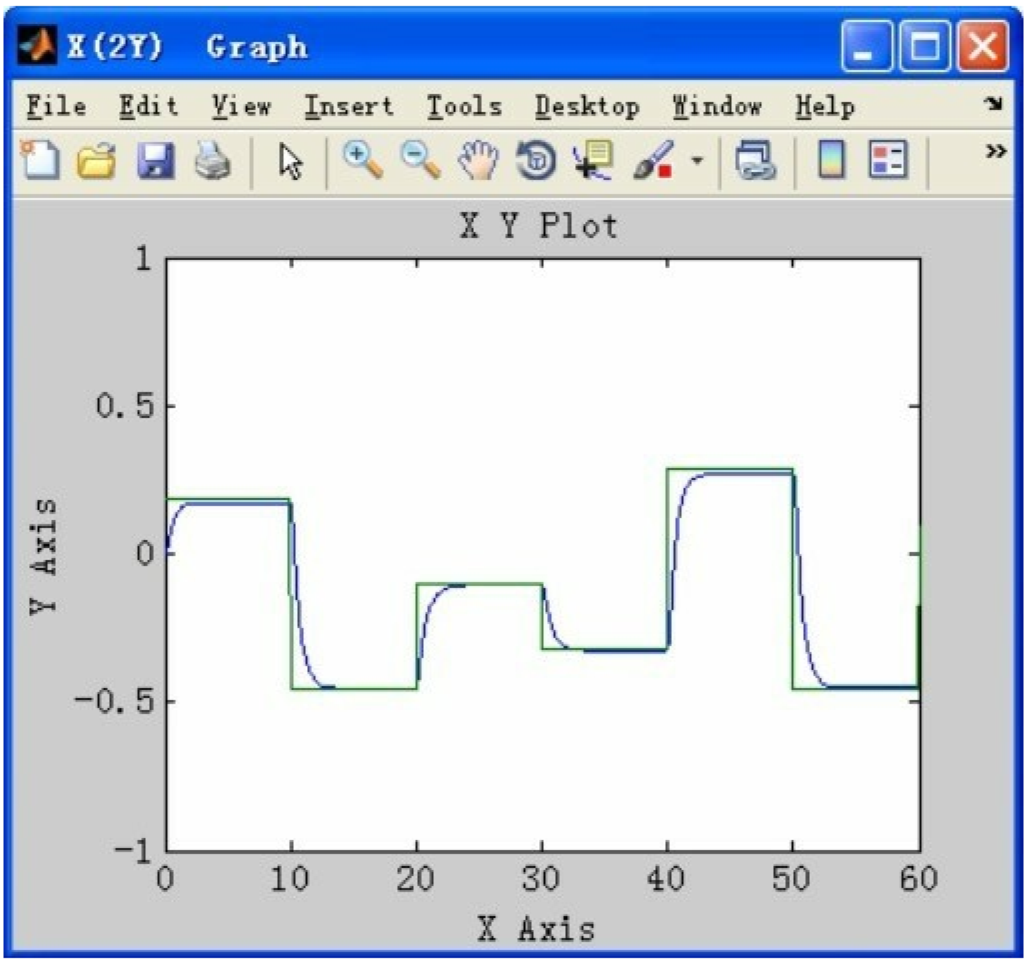


图11-43 系统的输出和参考信号

11.3 本章小结

Simulink称为系统动态仿真工具箱，其中Simulink神经网络仿真模型库是Simulink众多模型库中的一个重要部分。本章主要介绍了Simulink神经网络模型的类别及使用方法。对于神经网络控制系统，本章也对常用的神经网络模型预测控制、反馈线性化控制和模型参考控制这3种控制模型进行了介绍。

第12章 神经网络GUI

Simulink 神经网络工具箱提供了GUI（Graphical User Interfaces，简称图形用户界面），从而使用户在图形界面上，通过与计算机的交互操作设计和仿真神经网络，使得神经网络的设计和仿真变得简单易学。

本章将重点介绍Simulink神经网络GUI的概念及其使用方法。

学习目标：

·熟悉神经网络GUI 的概念

·熟悉神经网络GUI 的分类

·掌握神经网络GUI 的使用

12.1 GUI简介

用户界面（或接口）是指人与机器（或程序）之间交互作用的工具和方法。如键盘、鼠标、跟踪球、话筒等都可成为与计算机交换信息的接口。

GUI则是由窗口、光标、按键、菜单、文字说明等对象（Objects）构成的一个用户界面。用户通过一定的方法（如鼠标或键盘）选择、激活这些图形对象，使计算机产生某种动作或变化，比如实现计算、绘图等。

12.1.1 GUI设计工具

GUI设计工具包括以下几个。

·对象设计编辑器：在图形对象内创建和安排各种对象。

·菜单编辑器：创建、设置或修改下拉式菜单和内容式菜单。

·对象属性查看器：可查看每个对象的属性值，也可修改和设置对象的属性值。

·位置调整工具：可左右、上下对多个对象的位置进行调整。

·对象浏览器：可观察当前设计阶段的各个句柄图形对象。

1.对象设计编辑器

在MATLAB命令窗口中，选择New菜单下的



，或者在命令窗口中命令行上输入guide，MATLAB 会出现GUIDE Quick Start 提示框，如12-1 所示。

MATLAB 为用户提供了4 个GUI 模式供用户选择，分别是Blank GUI（空白版）、GUI with Uicontrols（带有控件的模板）、GUI with

Axes and Menu（带有坐标轴和菜单的模板）和Modal Questions Dialog（问答式对话框）。一般选择Blank GUI（即默认模式）进入对象设计编辑器，如图12-2所示。

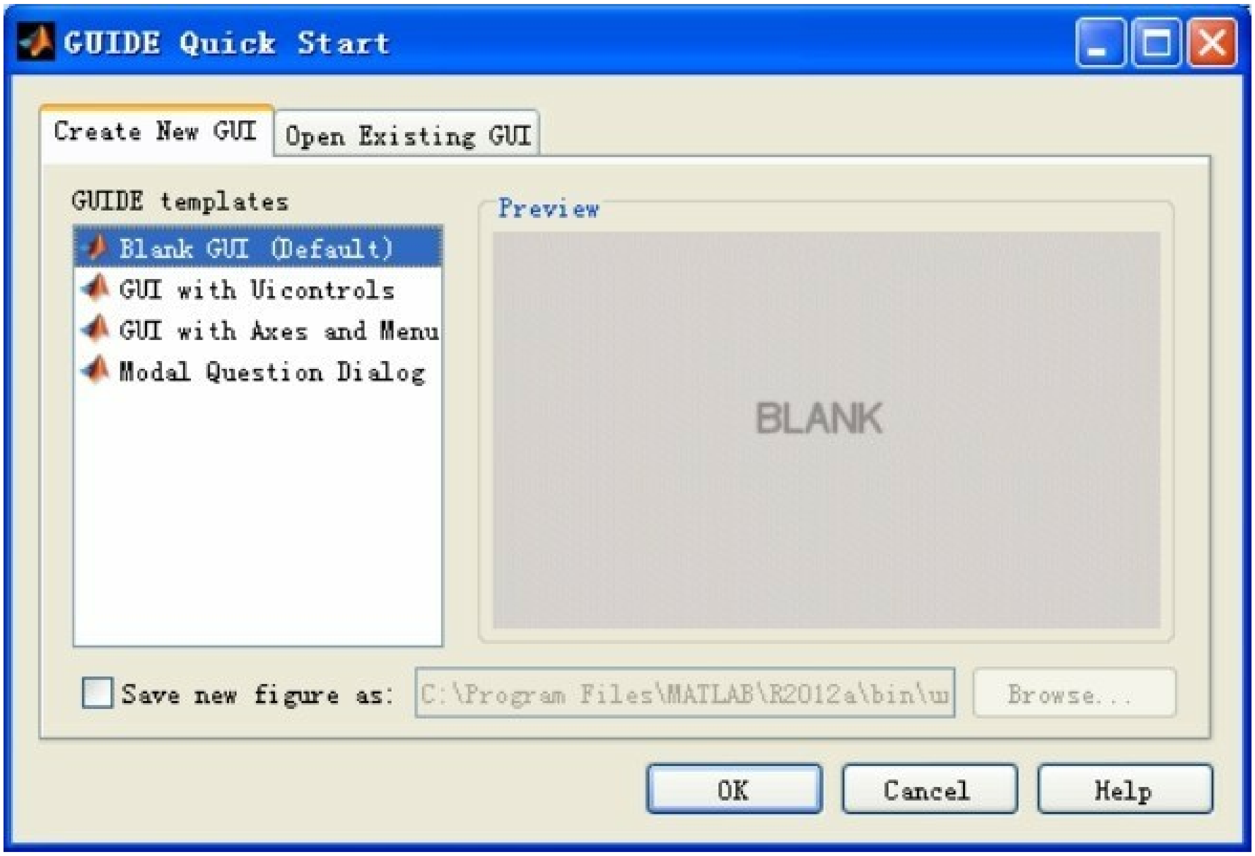


图12-1 GUIDE Quick Start提示框

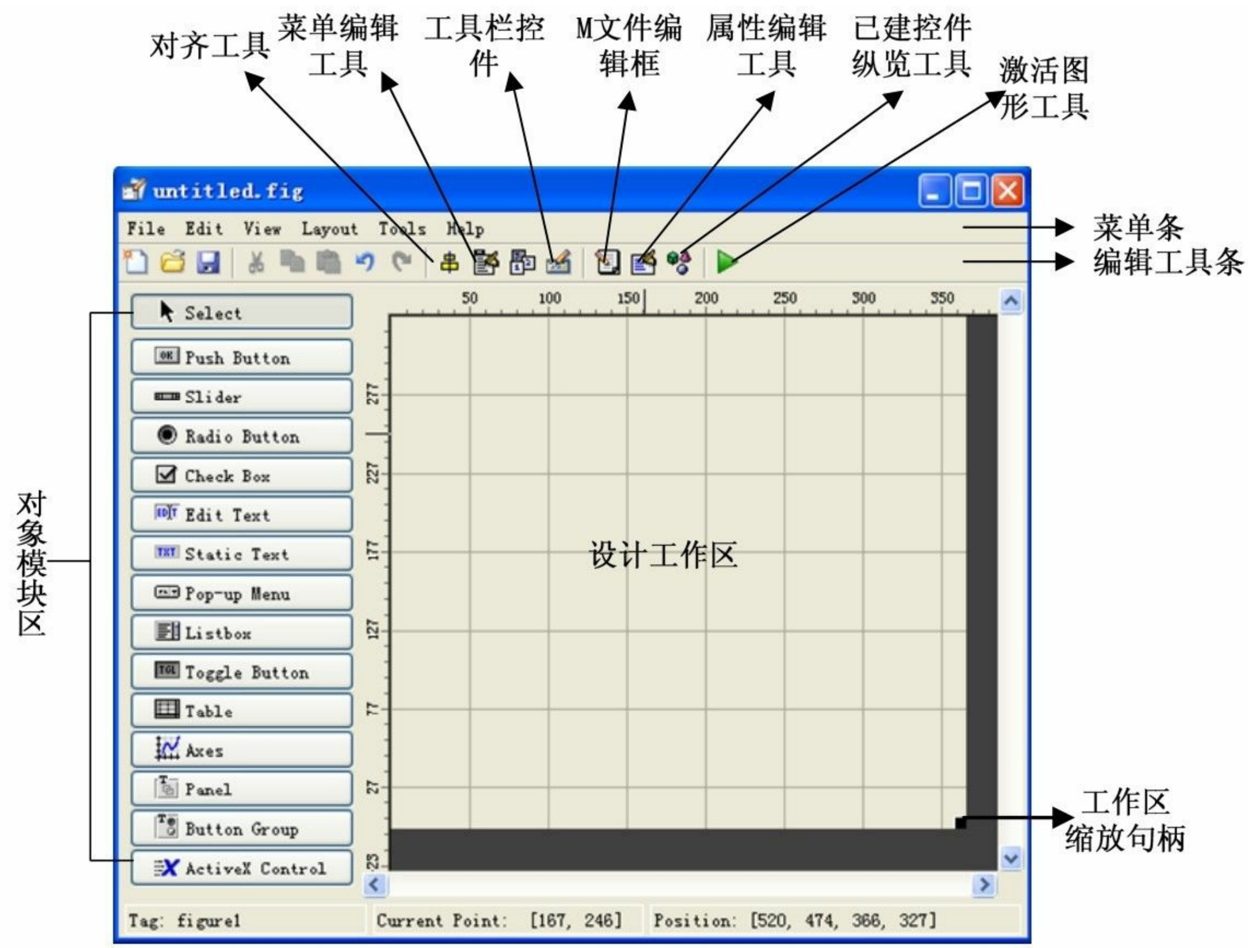


图12-2 对象设计编辑器

选择图12-2中左边某一控件拖到中间的对象设计区，即生成该对象，创建图形对象后，双击该对象，就会显示该对象的属性编辑器，如图12-3所示。

在对象设计编辑器界面的工具条上，有菜单编辑器、位置调整器、属性编辑器、对象浏览器的按钮，可用其方便地调用需要使用的GUI设计工具。

2.菜单编辑器

利用菜单编辑器，可以创建、设置和修改下拉式菜单和内容式菜单。单击 Tools-menu editor进入菜单编辑器，菜单编辑器左上角的按钮用于添加或删除下拉式菜单和子菜单，以及对菜单的设置和修改。

具体如图12-4所示。

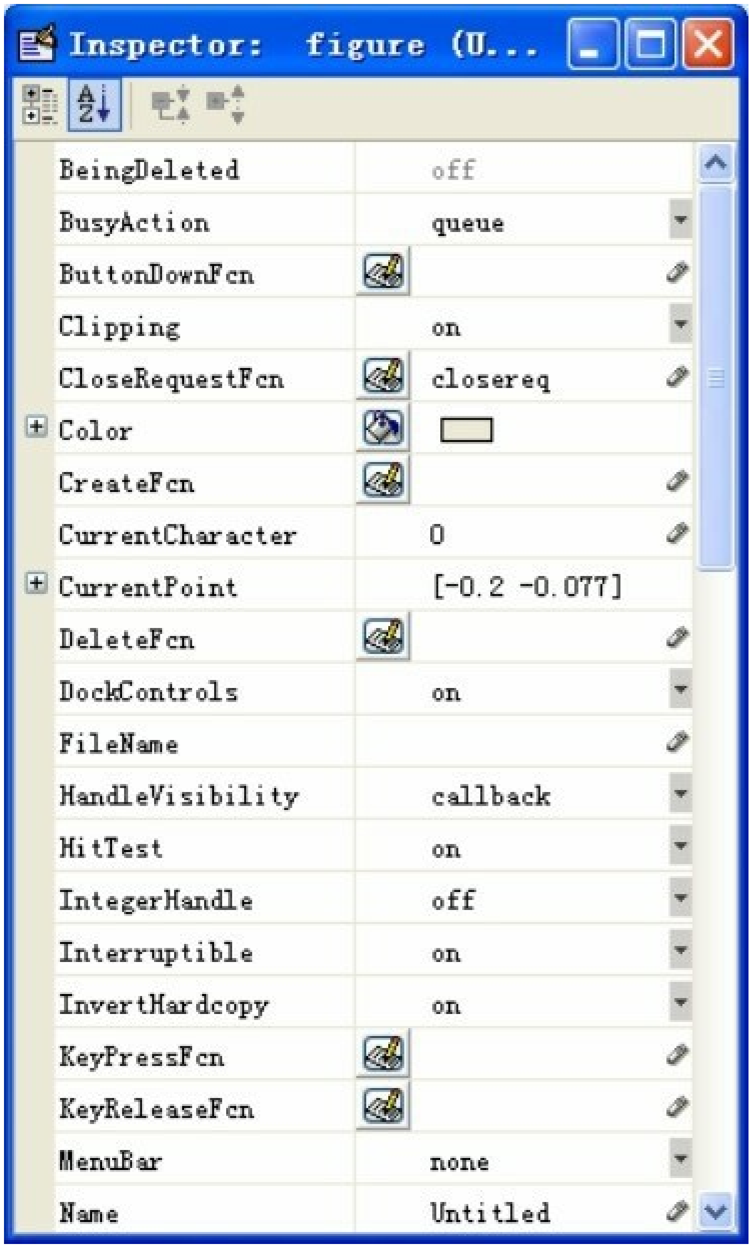


图12-3 对象属性编辑器

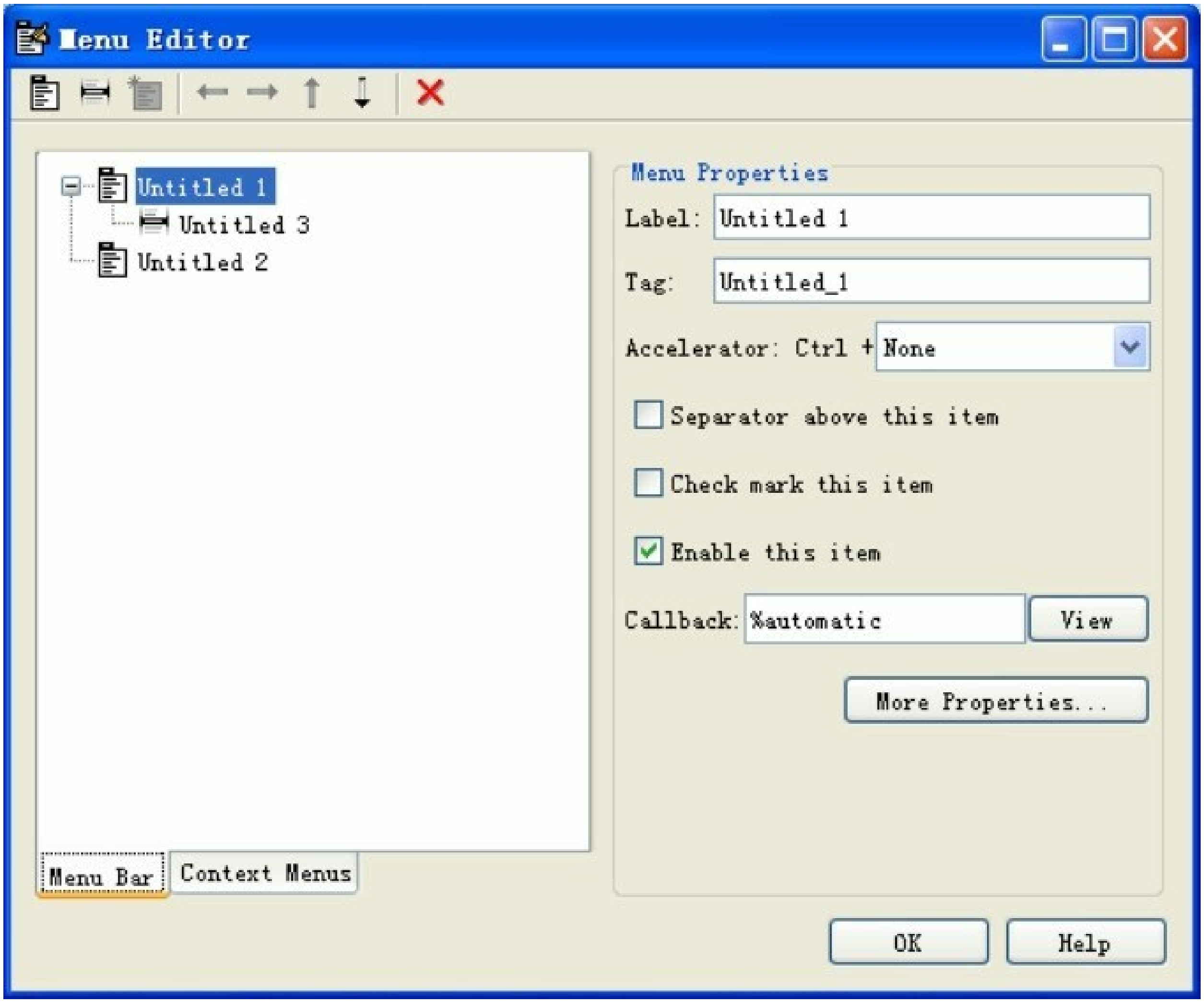


图12-4 对象属性查看器

3.位置调整工具

利用位置调整工具，可以对对象设计编辑区中的多个对象的位置进行调整。选择工具条上“Align Objects”按钮，或者选择Layout 菜单下Align Objects子菜单，就可看到对象位置调整器界面，如图12-5所示。

第一栏用于垂直方向的位置调整，Align 表示对象间垂直对齐，Distribute 表示对象间垂直距离。在选中 Distribute 中某个按钮后，“Set Spacing”变为可用，即可设置对象间距离，其中距离单位为像素（Pixels）。

第二栏用于水平方向的位置调整，可以同样的方法调整水平位置。

4.对象浏览器

利用对象浏览器，可查看当前设计阶段的各个句柄图形对象。从对象设计编辑器界面工具条上选择“Object Browser”按钮，或者选择Tool菜单下Object Browser子菜单。就可以进入对象浏览器界面。

在对象设计编辑器界面上，要编制某一控件的回调程序，即在该控件上单击右键，在弹出菜单中选择ViewCallbacks子菜单，从中选择一种激活回调程序的方式，就可以编制程序了，如 CreateFcn指控件创建时调用该程序。

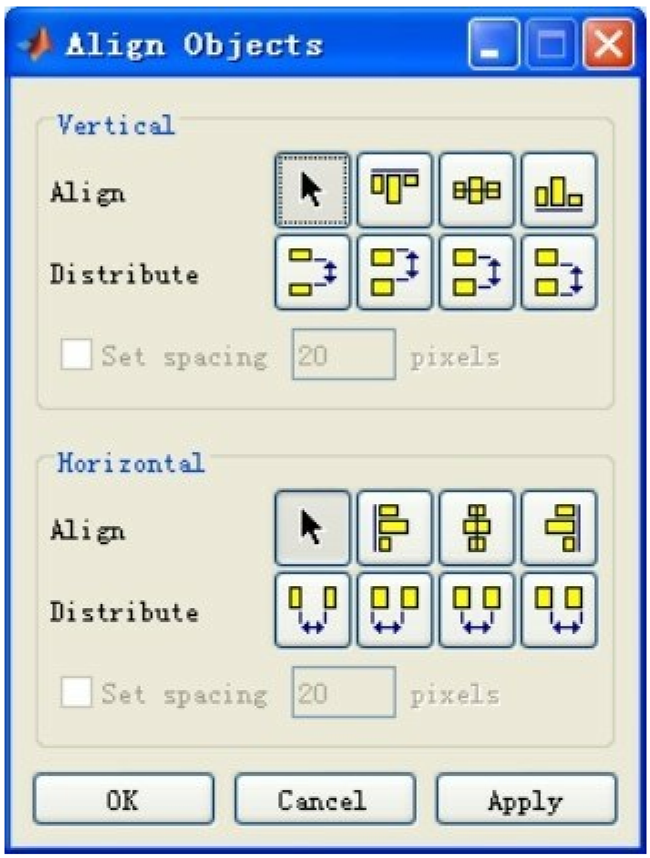


图12-5 位置调整工具

12.1.2 启动GUIDE

在本小节中，将使用GUIDE来定制GUI对象的界面，使用M文件来编写对应的事情程序，完成整个GUI的创建工作。GUI参数设置模板如

图12-6所示。

在GUI对象中，可以在文本框中输入三维函数的分量，绘制该三维函数的向量场和旋度场图形，同时可以选择输出计算的结果。

下面分步骤来介绍如何创建该GUI对象。

【例12-1】在MATLAB中，创建上面的GUI对象。

解：首先启动GUIDE。选择MATLAB工作界面上菜单栏中的“GUIDE”快捷菜单，启动MATLAB中的GUIDE。

打开空白板。在GUIDE Quick Start 对话框中选择“Blank GUI（Default）”选项，单击“OK”选项，MATLAB会显示GUIDE初始化对话框。

当MATLAB完成GUIDE初始化之后，就会显示GUI的空白模板，如图12-7所示。

设置模板的显示属性。选择空白模板菜单栏中File-Preferences选项，打开Preferences对话框，然后在其中选择“GUI”选项，选中“Show names in component palette”选项，显示空间面板中各个空间的名称。

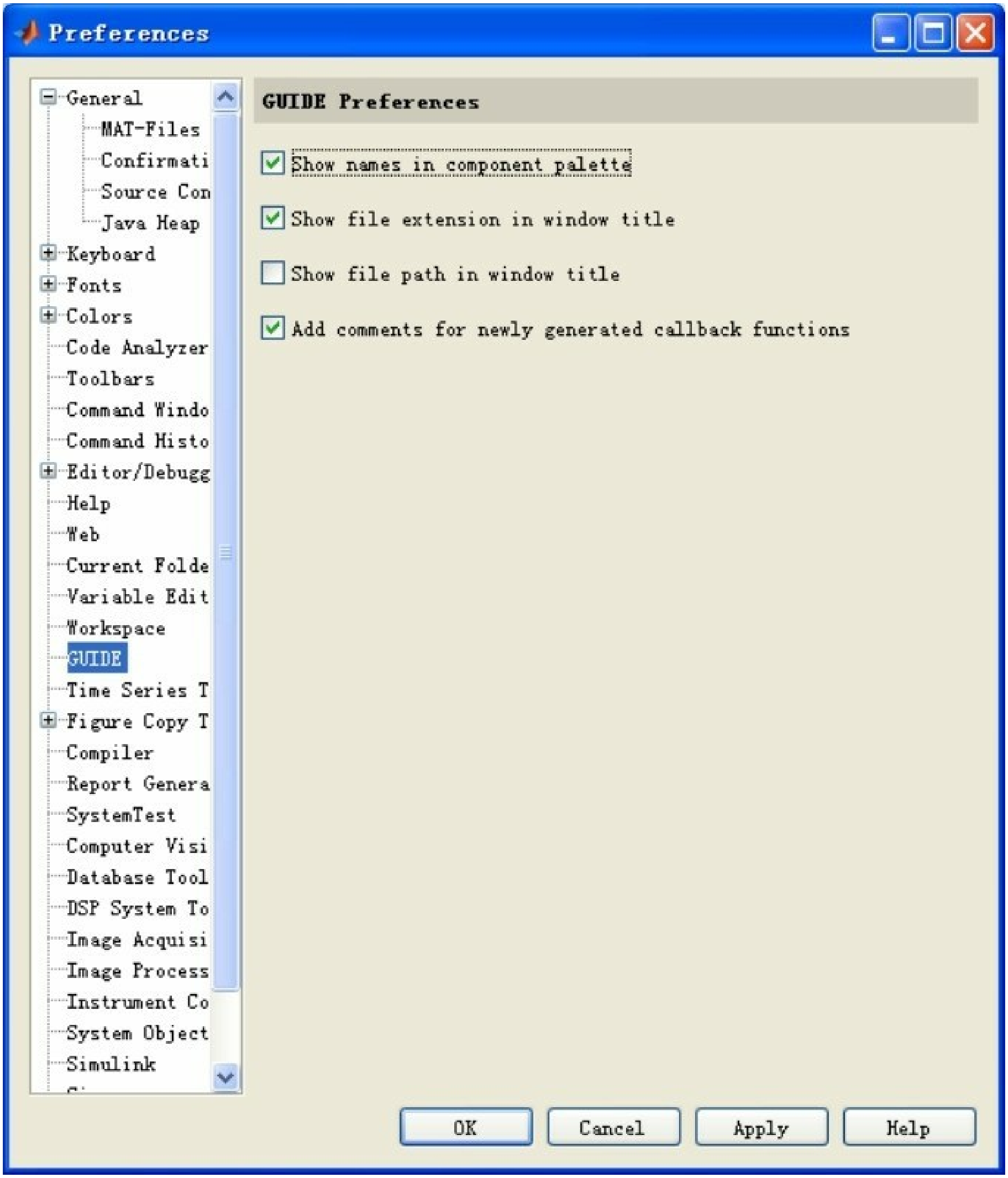

Preferences
General
MAT-Files
Confirmati
Source Con
Java Heap
Keyboard
Fonts
Colors
Code Analyzer
Toolbars
Command Windo
Command Histo
Editor/Debugg
Help
Web
Current Folde
Variable Edit
Workspace
GUIDE
Time Series T
Figure Copy T
Compiler
Report Genera
SystemTest
Computer Visi
Database Tool
DSP System To
Image Acquisi
Image Process
Instrument Co
System Object
Simulink
GUIDE Preferences
Show names in component palette
Show file extension in window title
Show file path in window title
Add comments for newly generated callback functions
OK
Cancel
Apply
Help

图12-6 参数设置模板

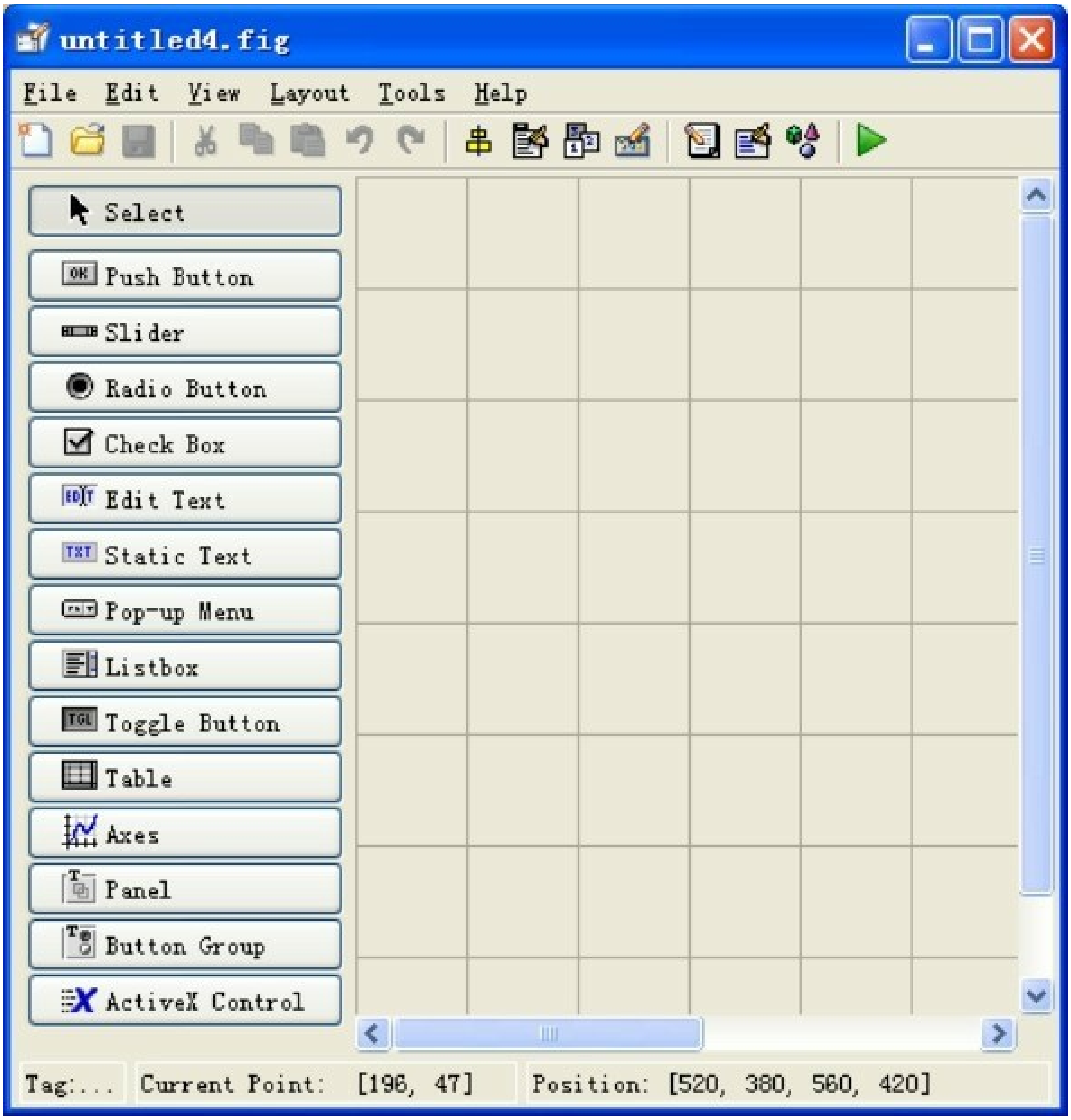


图12-7 修改后的空白模板

对于那些首次接触MATLAB的GUI设计过程的读者来说，需要显示空间面板中各个控件名称，这样可以帮助他们熟悉相应的控件。

12.1.3 添加控件组件

延续上面小节的步骤。

向面板中添加“坐标轴”控件。从“控件面板”中选择“Axes”对象，将

其拖动到空白模板中合适的位置上，得到的结果如图12-8所示。

在MATLAB中，坐标轴控件的主要功能是使用户的GUI对象可以显示图像对象，用户可以为坐标轴控件设置外观和行为参数。

在本实例中，不需要为坐标轴控件设置参数，直接使用MATLAB中的默认属性就能很好地显示该实例总的所有图形对象。

复制坐标轴控件。选中上面步骤中添加的Axes控件，单击鼠标右键，如图12-9所示。

在弹出的快捷菜单中选择“Duplicate”选项，复制上面步骤添加的坐标轴控件，得到结果如图12-10所示。

添加组合框（panel）控件。在本实例中，需要添加绘制图形的各种参数，为了便于管理，需要将这些参数放置在组合框控件中。在控件面板中选择“Panel”控件，然后将其添加到面板中，如图12-11所示。

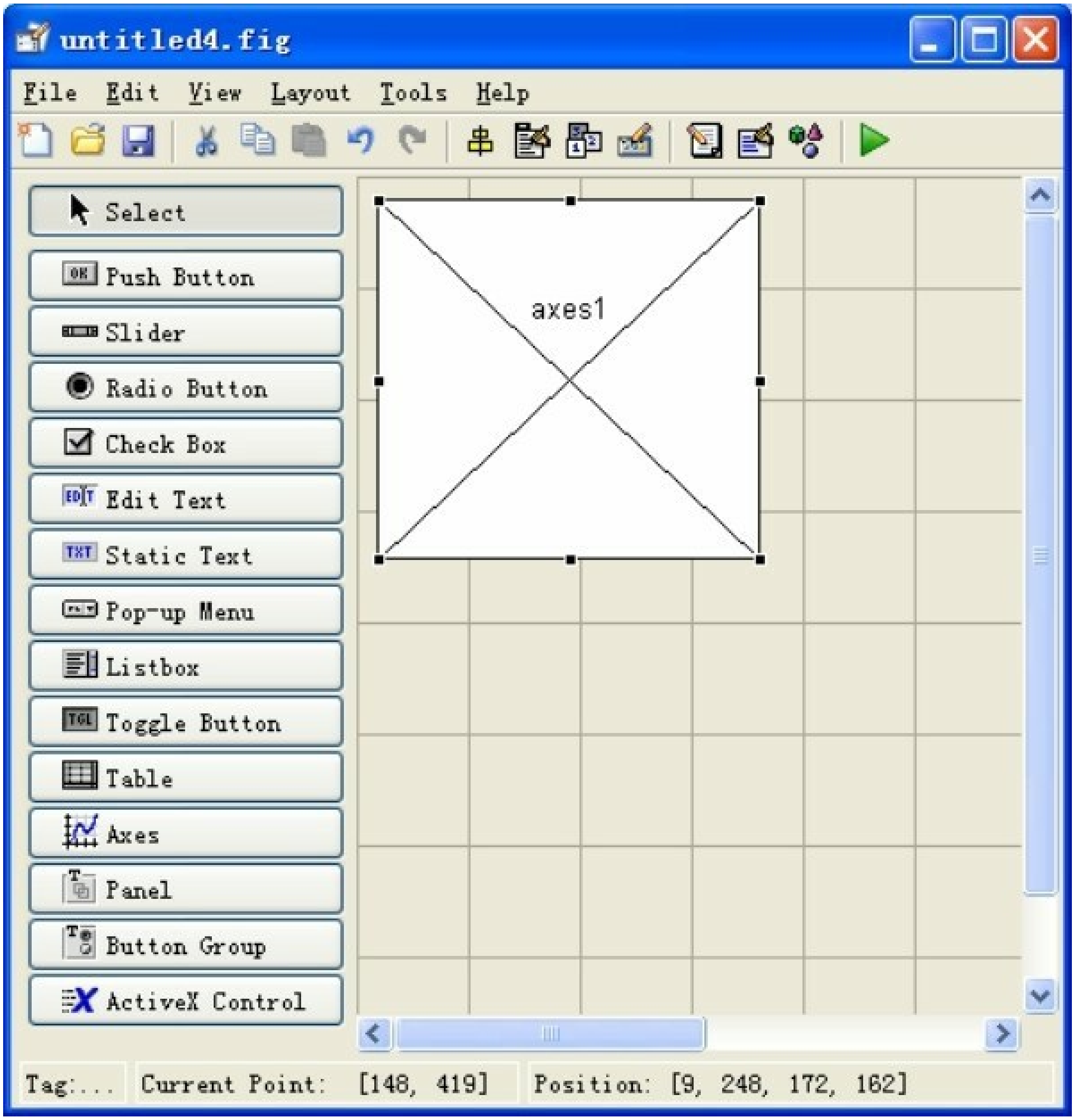


图12-8 添加坐标轴控件

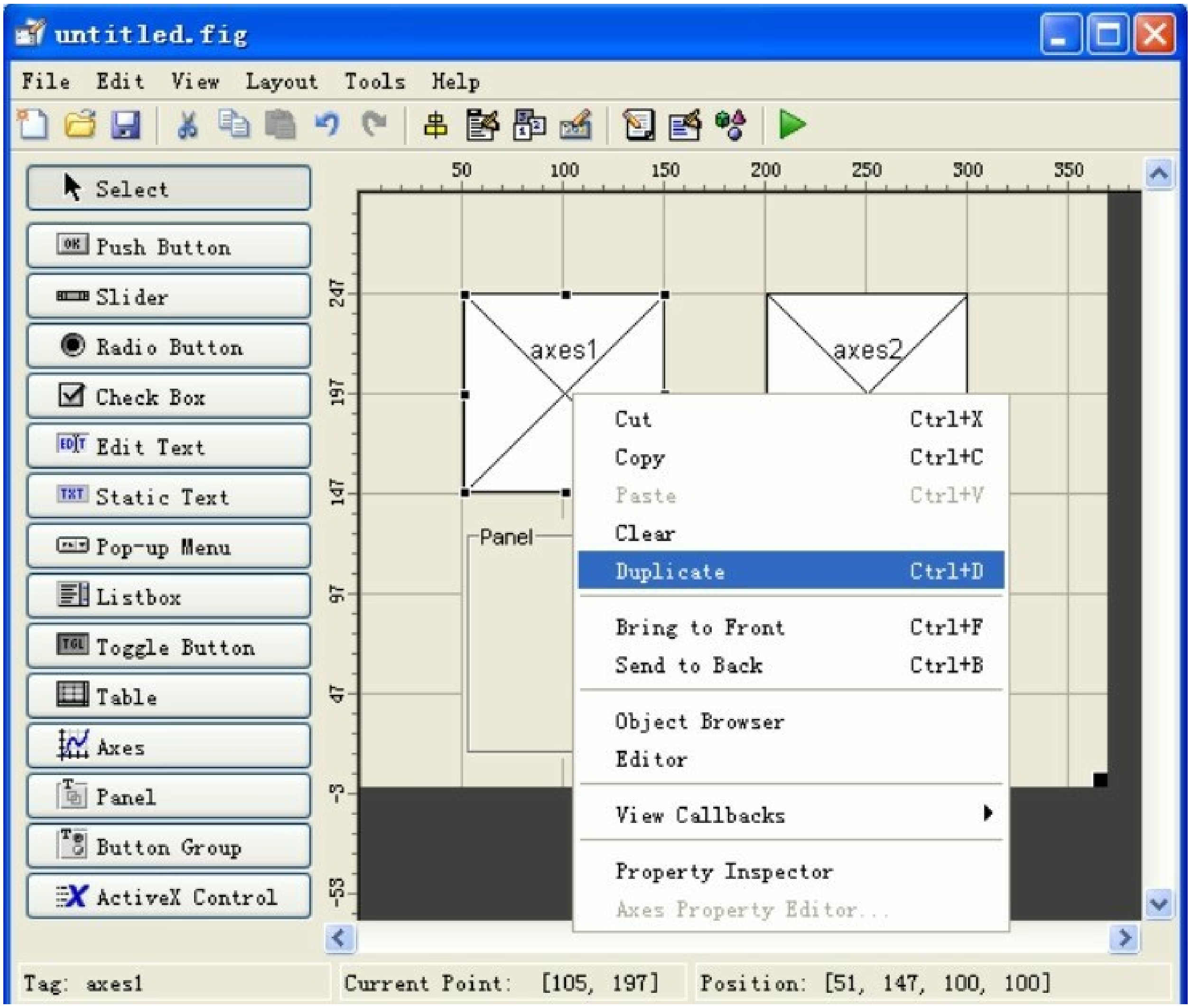


图12-9 复制坐标轴控件

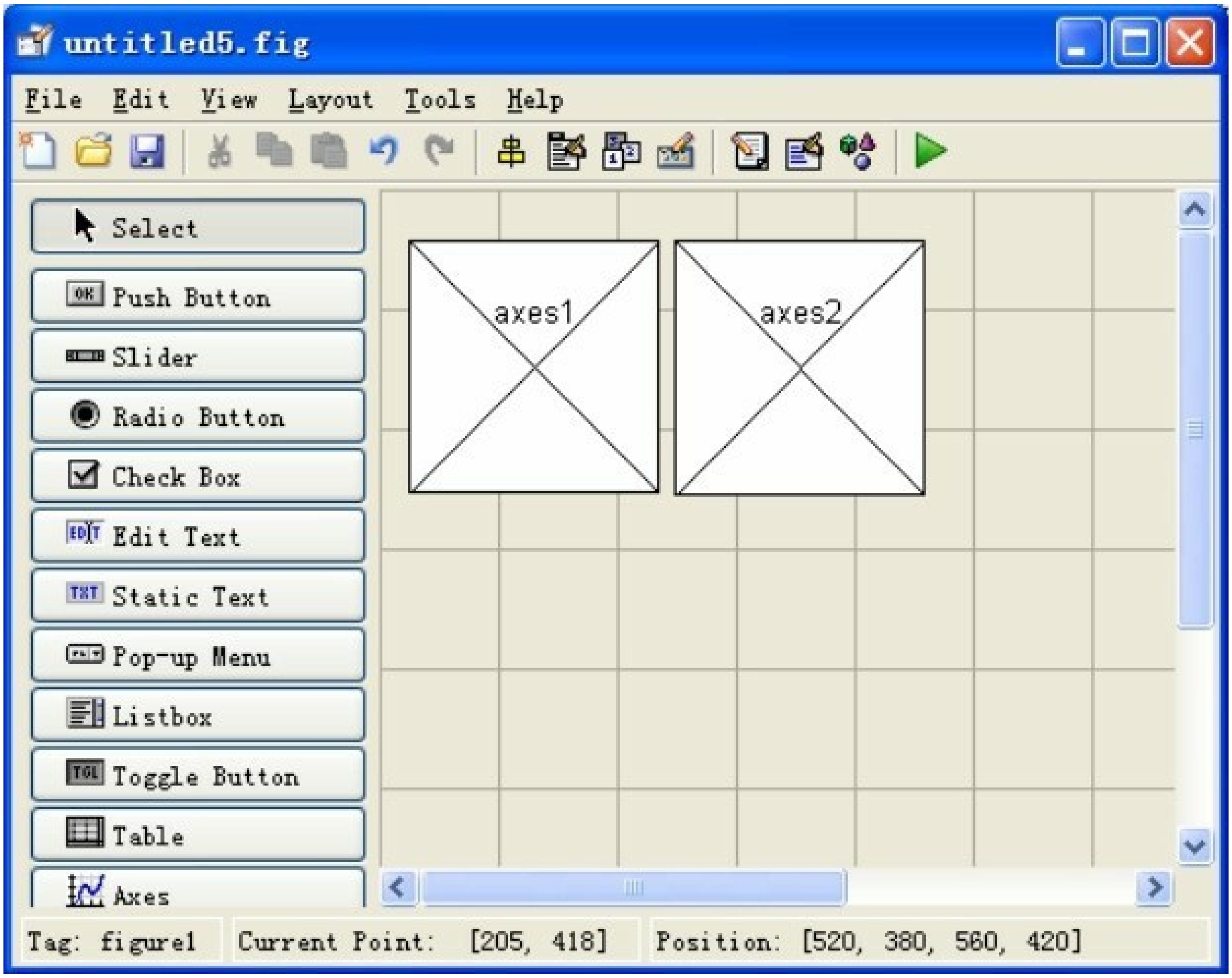


图12-10 复制后的坐标轴控件

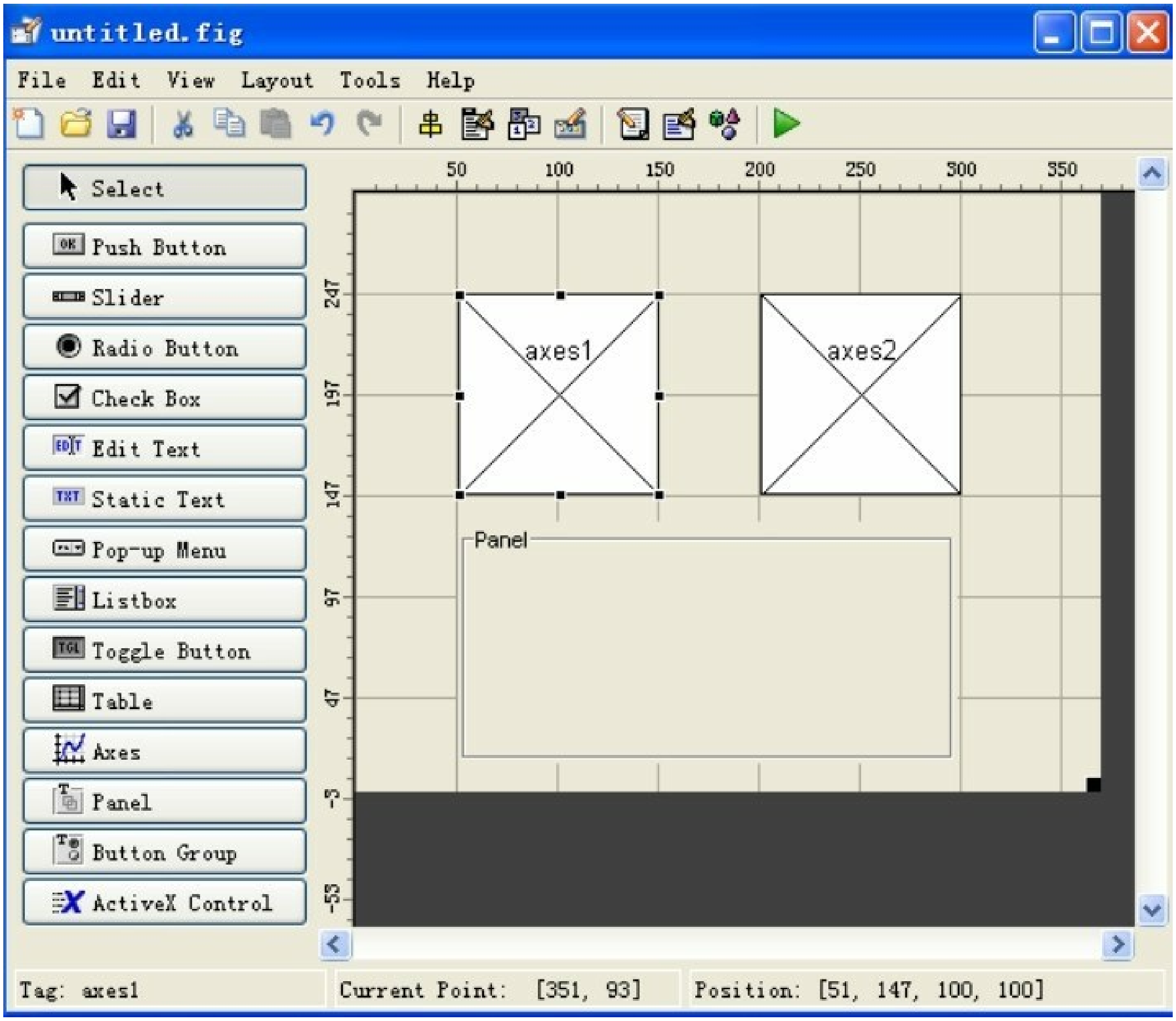


图12-11 Panel控件框

在MATLAB中，组合框是图形窗口中的一个封闭区域，它可以将相关联的控件（例如一组单选按钮或者一组编辑框等）组合在一起，使得图形窗口变得更加容易理解。该组合框的主要属性包含标题和边框。

在默认情况下，组合框的边框类型是“etchedin”，也就是内嵌的蚀刻效果；其标题为“Panel”，可以通过“属性检查器（Properfy Inspector）”来修改这些属性。

添加“编辑框（Edit Text）”控件对象。选择控件面板中的“Edit Text”对象，然后将其添加到上面步骤中添加的“Panel”对象中，如图12-12所示。

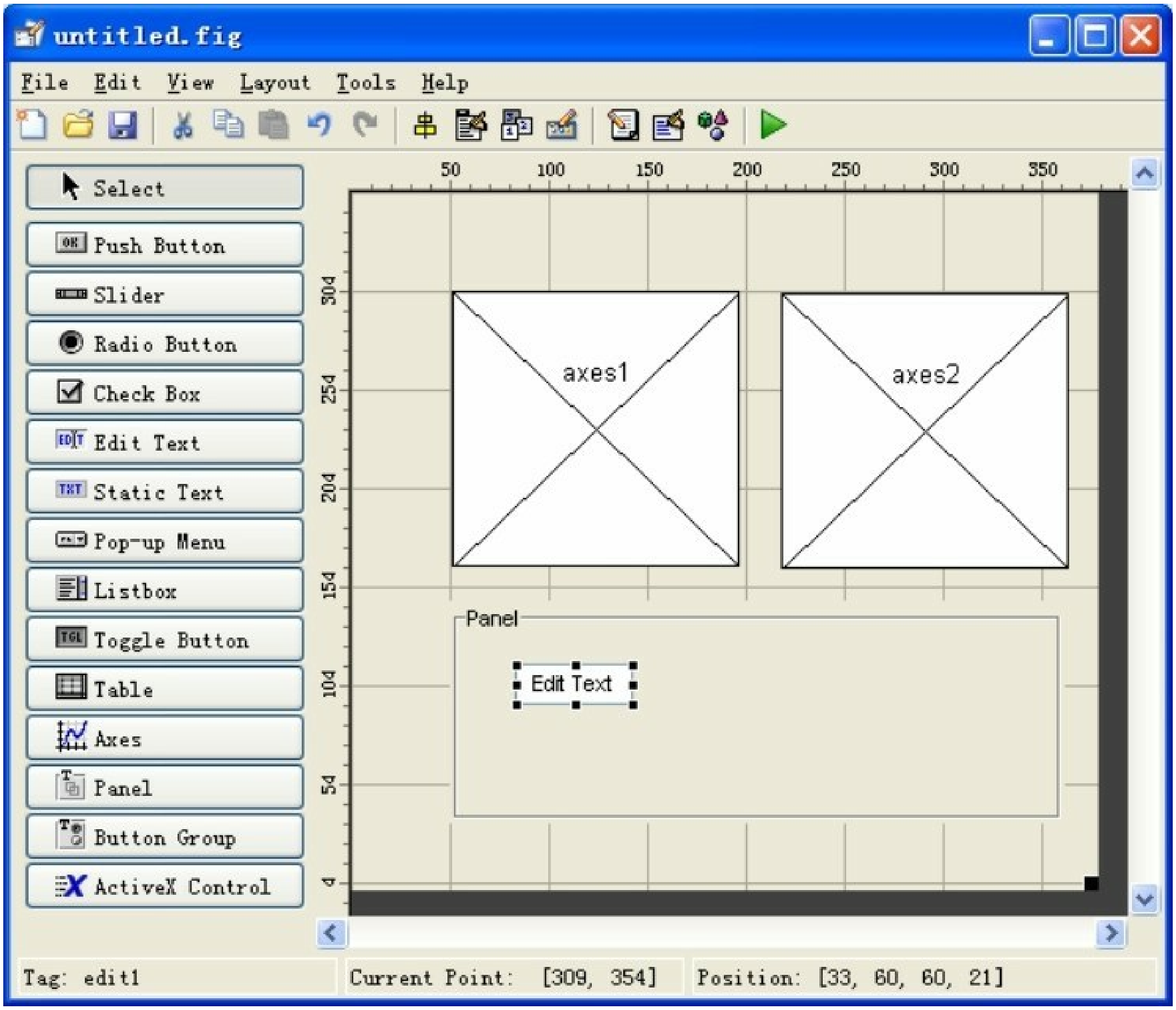


图12-12 添加编辑框控件

在MATLAB中，编辑框控件的功能是控制用户编辑或者修改字符串的文本区域，该对象的String属性包含了用户输入的文本信息。在Windows操作系统中，单击图形窗口不会引发编辑框的回调函数的执行。

添加“静态文本（Static Text）”控件对象。选择控件面板中的“Static Text”对象，然后将其添加到上面步骤中添加的“Panel”对象中，如图12-13所示。

在MATLAB中，静态文本控件的功能是用作其他控件的标签，该控件对象和编辑框控件的主要区别在于，用户不能在交互的状态下修改

其文字或者调用其相应的回调函数。在本实例中，该控件的功能在于显示输入参数的标签。

复制“编辑框（Edit Text）”和“静态文本（Static Text）”控件对象。使用前面步骤介绍的方法，复制添加的“编辑框”和“静态文本”控件，结果如图12-14所示。

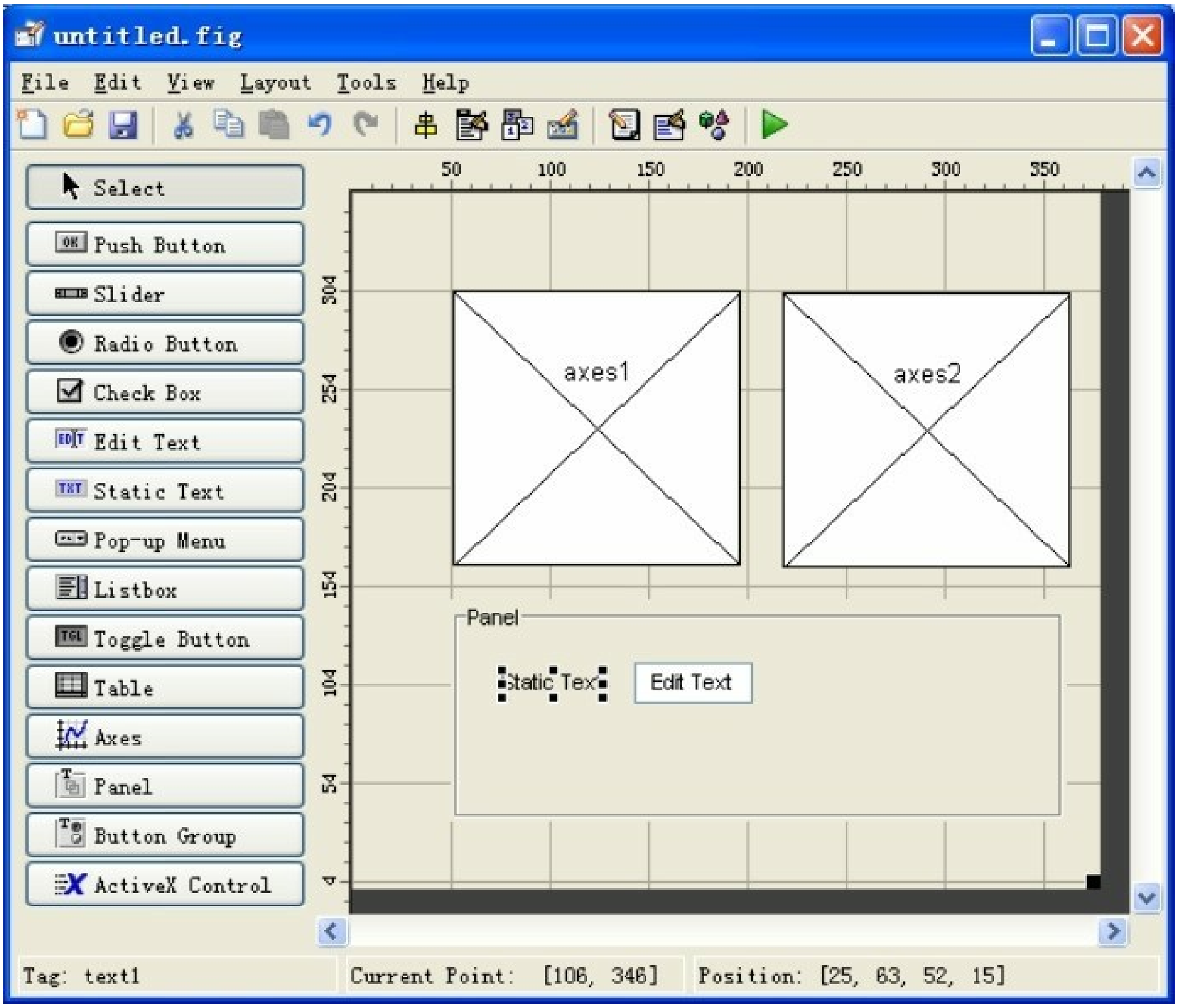


图12-13 添加静态文本控件对象

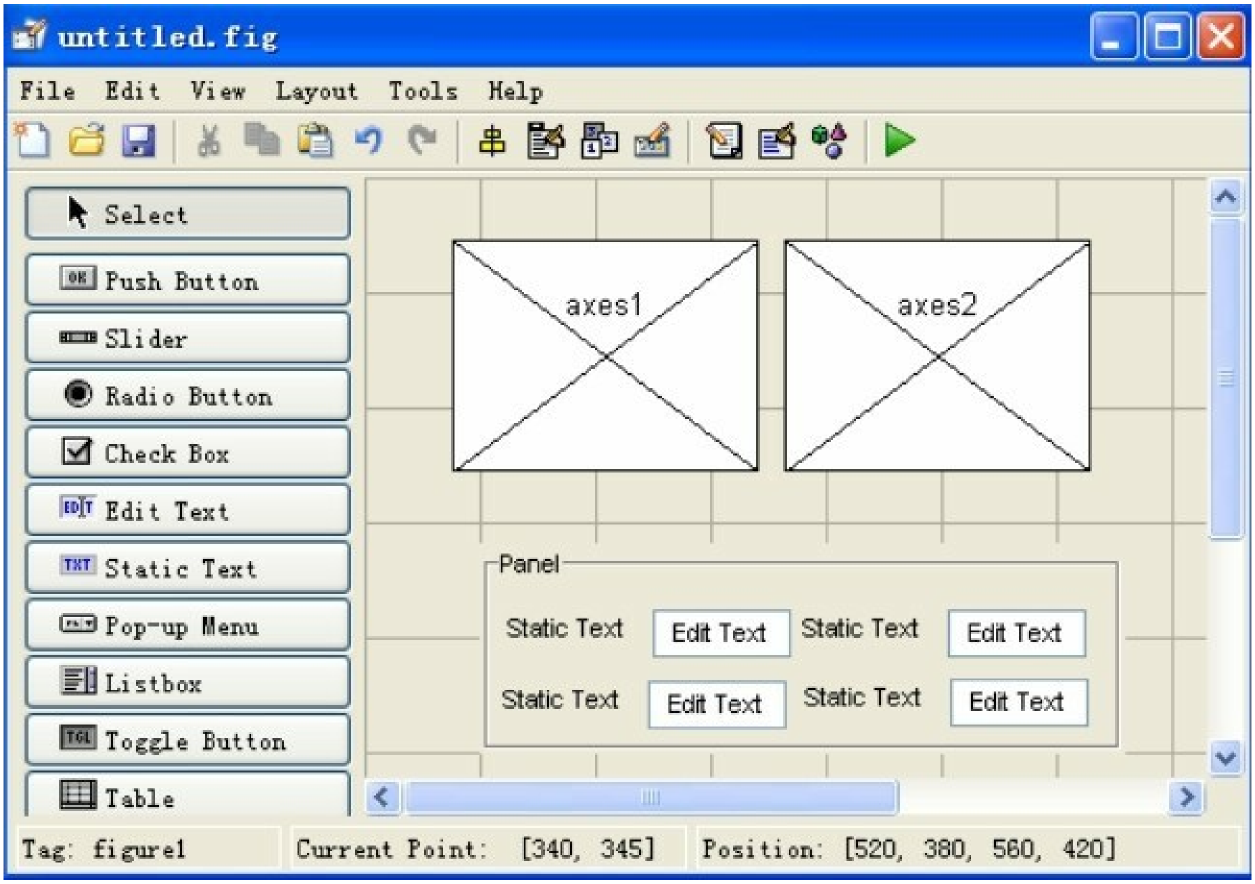


图12-14 复制编辑框和静态文本

添加“按钮（Push Button）”控件。选择控件面板中的“Push Button”对象，将其添加到面板中，如图12-15所示。

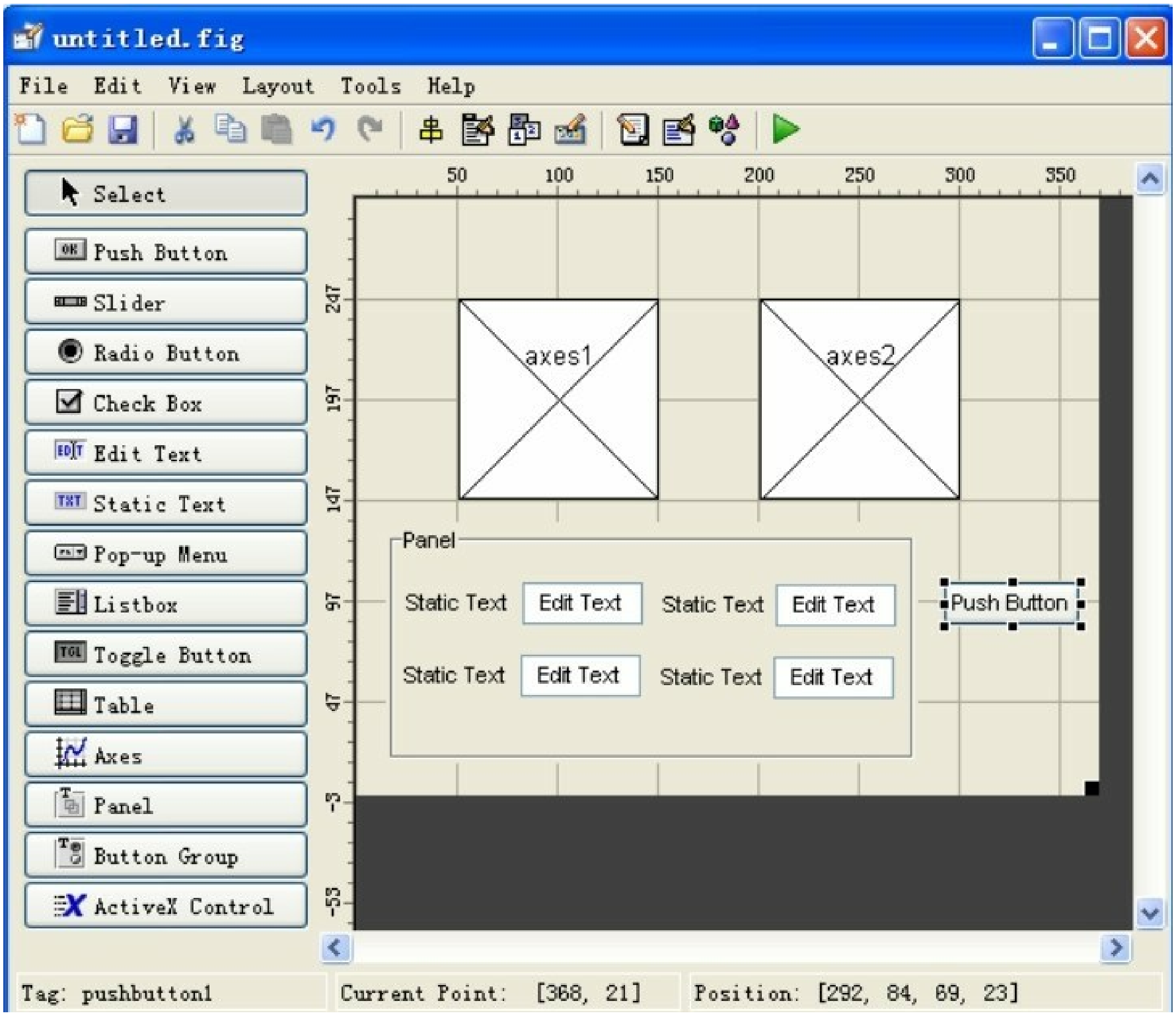


图12-15 添加按钮控件

和其他编程实例类似，按钮是实现用户和程序互动的主要控件类型，通过鼠标单击按钮来实现某种行为并调用相应的回调函数。

复制上面步骤的“Push Button（按钮）”控件。由于本实例中图形界面中包含两个按钮控件，因此在本步骤中需要复制该按钮控件，如图12-16所示。

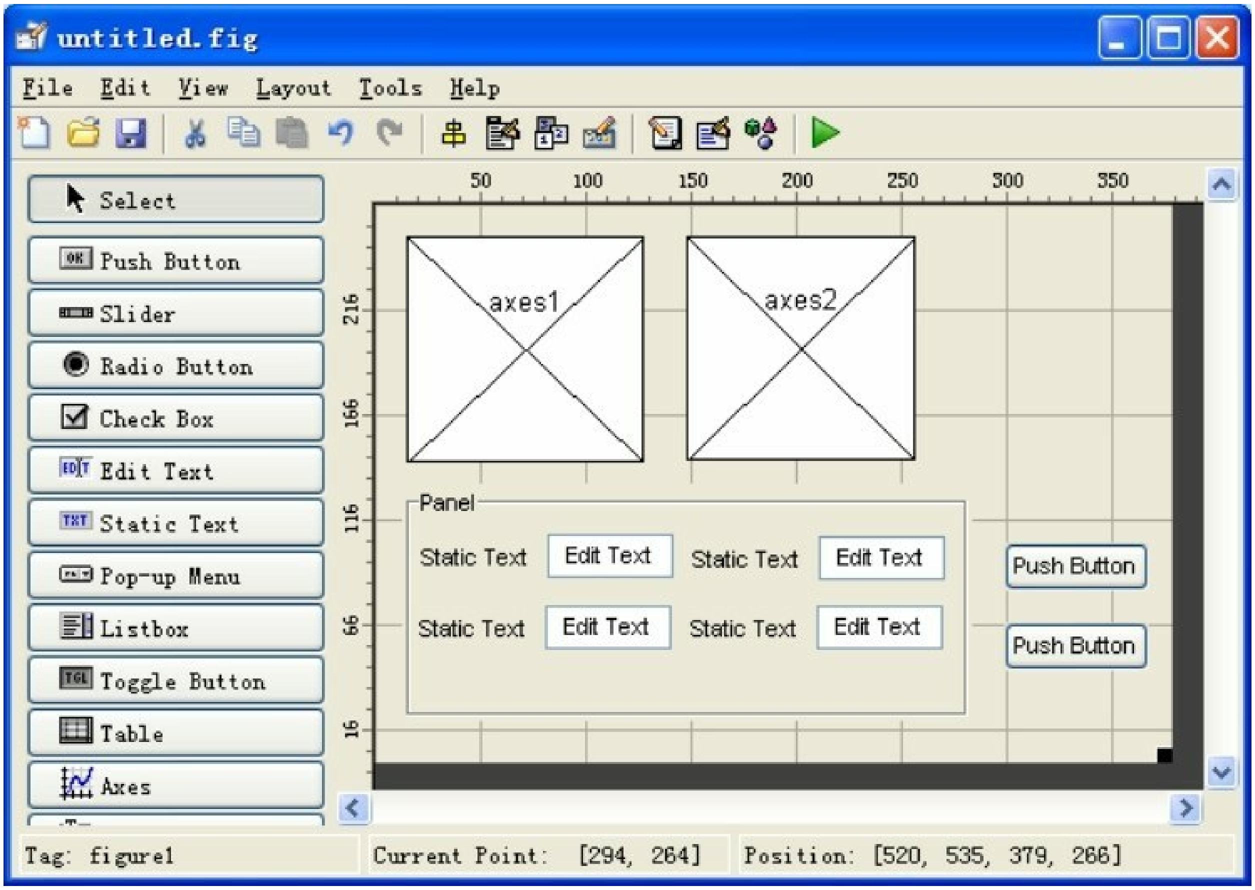


图12-16 复制按钮控件

12.1.4 设置控件组件的属性

延续上面小节的步骤。

设置图形界面的标题属性。选中整个图形界面，单击“Property Inspector”按钮，打开Property Inspector 对话框，然后选择“Name”选项，在其中输入标题“Simulator”，如图12-17所示。

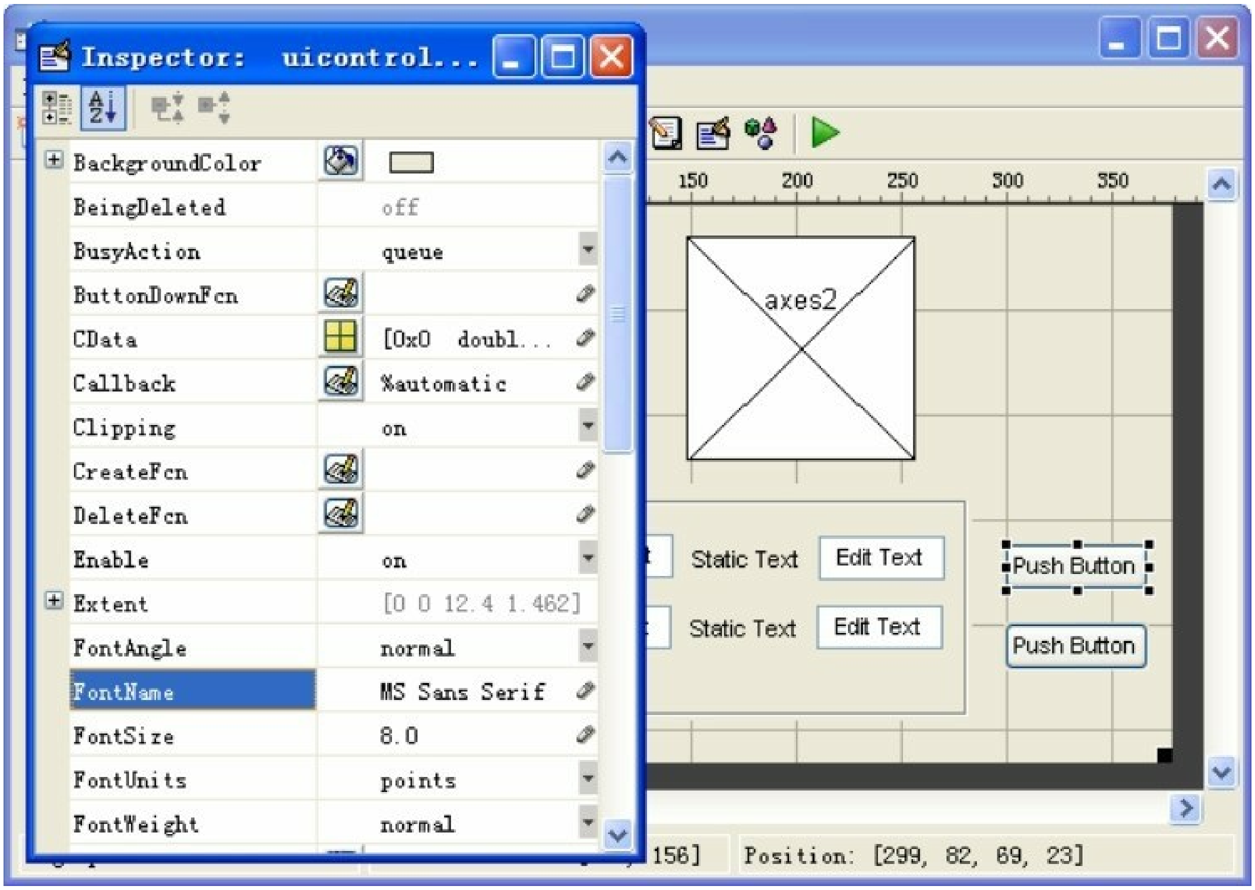


图12-17 设置图形界面的标题

设置组合框控件的属性。选中“Panel”控件，单击“Property Inspector”按钮，打开Property Inspector对话框，在其中设置组合框控件的属性。

在上面的Property Inspector 对话框中，选择“HighlightColor”选项，将其设置为空白，其他属性保存系统默认属性。

设置“静态文本”控件的属性。选择第一个“静态文本”控件，单击“Property Inspector”按钮，打开Property Inspector 对话框，选择“String”选项，在其中输入“X”，得到的结果如图12-18所示。

依次修改其他“静态文本”控件的文字。可以使用和上面步骤类似的方法来修改其他“静态文本”控件的文字，修改后的结果如图12-19所示。

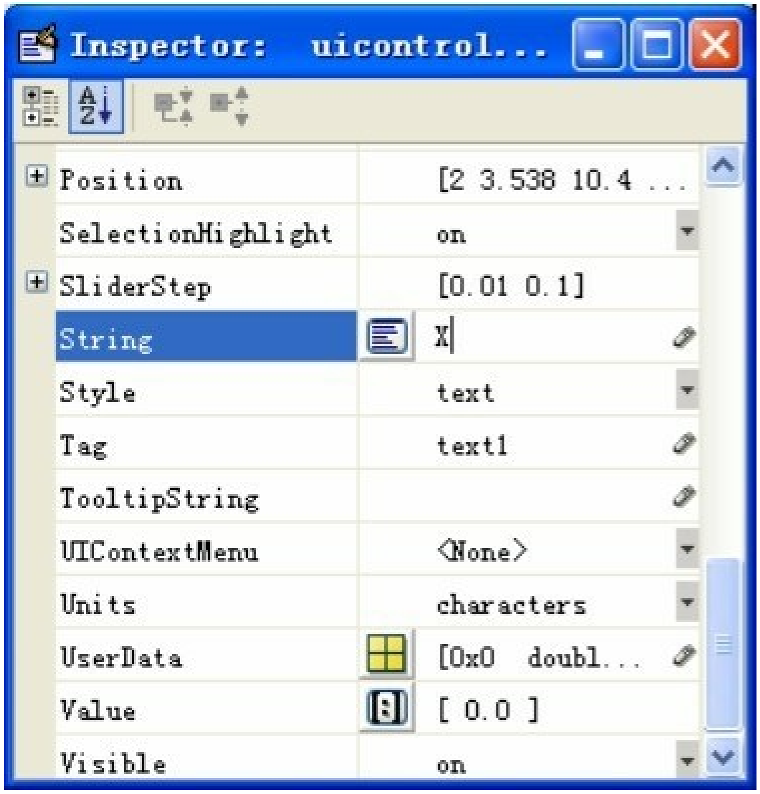


图12-18 设置“静态文本”的标题

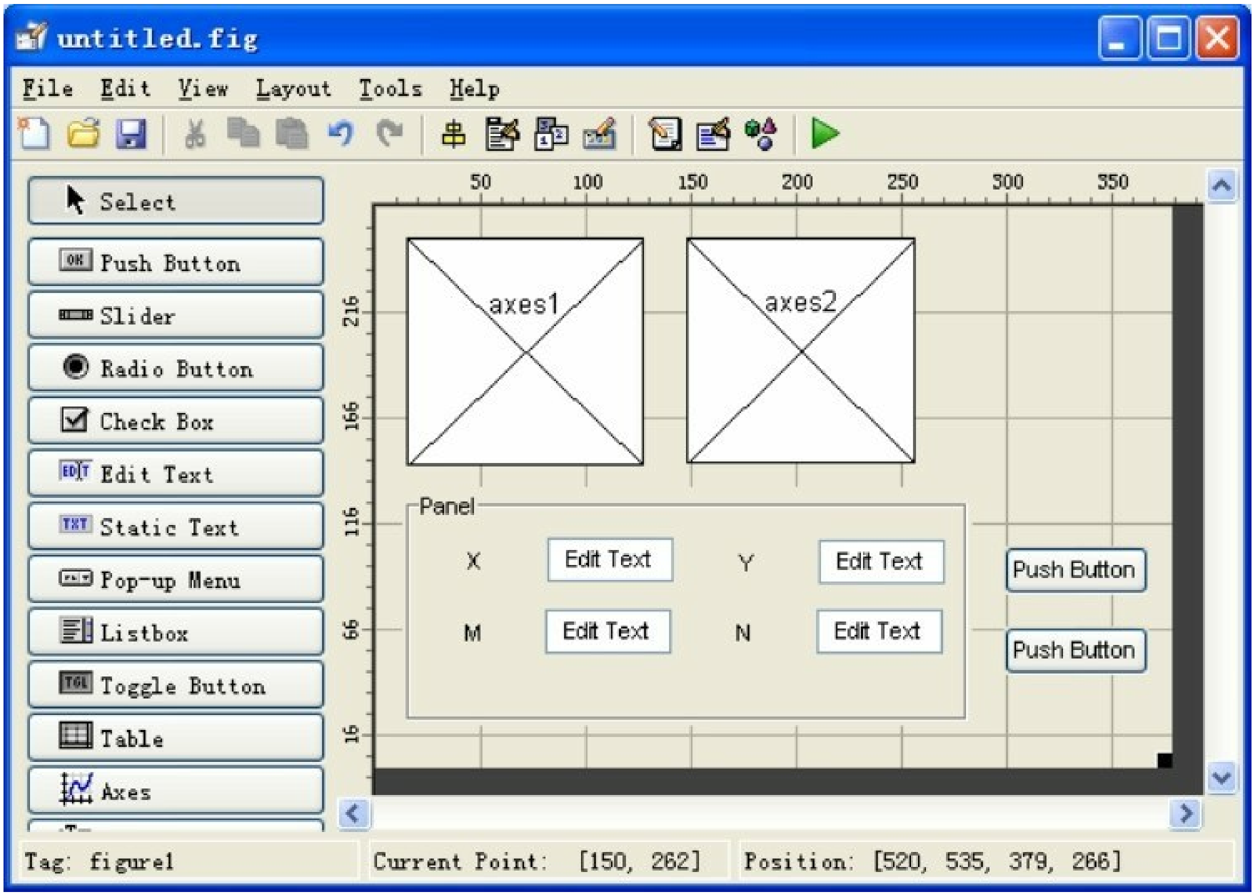


图12-19 修改其他“静态文本”控件的文字

设置左侧“编辑框”控件的属性。在本实例中，左侧的两个编辑框提供给用户输入分量表达式，因此该编辑框的默认数值是“空格”。选择第一个“编辑框”控件，然后单击“Property Inspector”按钮，打开Property Inspector对话框，选择“String”选项，将其设置为“空”，得到的结果如图12-20所示。

设置右侧“编辑框”控件属性。选择右侧第一个“编辑框”控件，单击“Property Inspector”按钮，打开Property Inspector 对话框，选择“String”选项，将其设置为“[-1,1]”，得到的结果如图12-21所示。

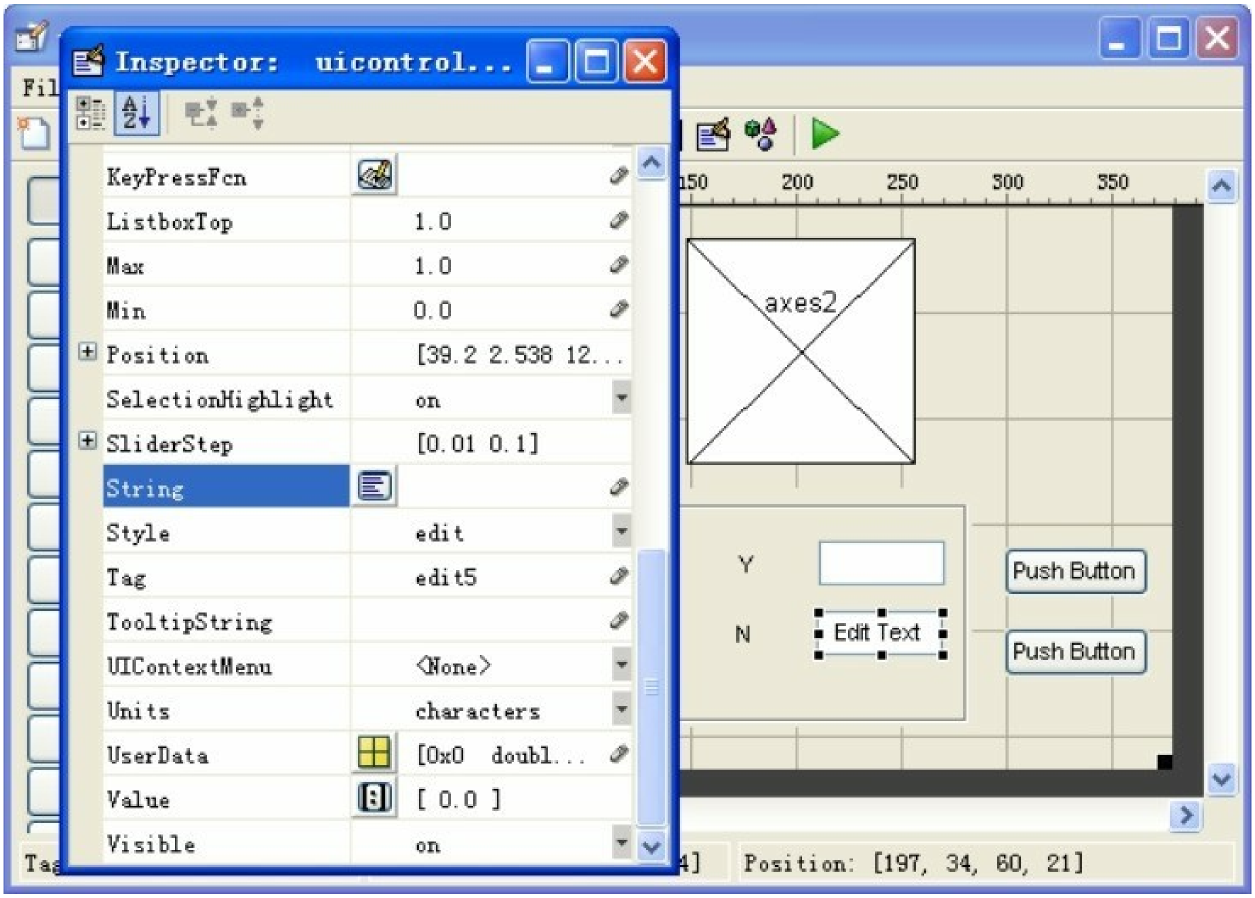


图12-20 设置左侧编辑框控件的属性

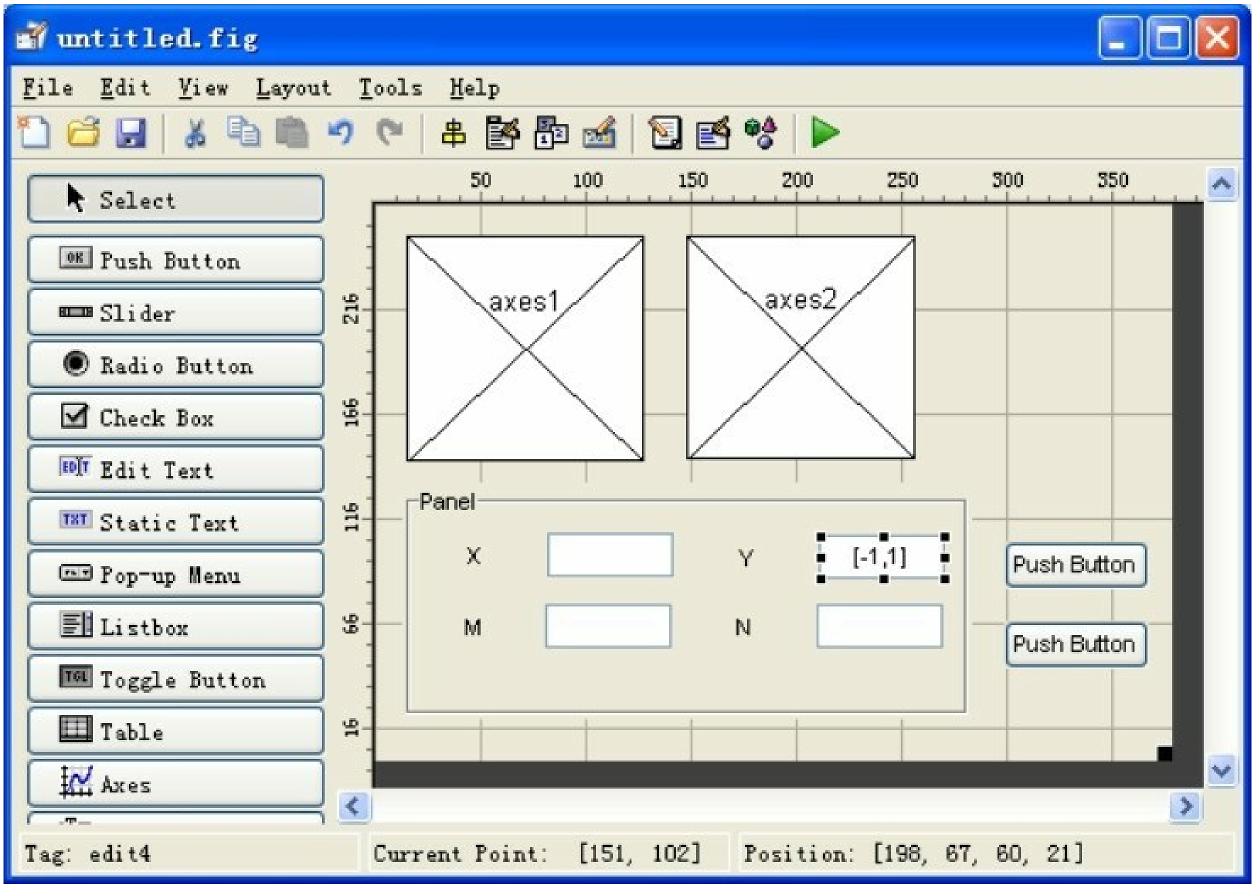


图12-21 设置右侧编辑框控件的属性

查看编辑后的控件。使用相同的方法设置其他编辑框的属性，然后适当编辑其他编辑框的宽度，以及对齐方式，得到的最后结果如图12-22所示。

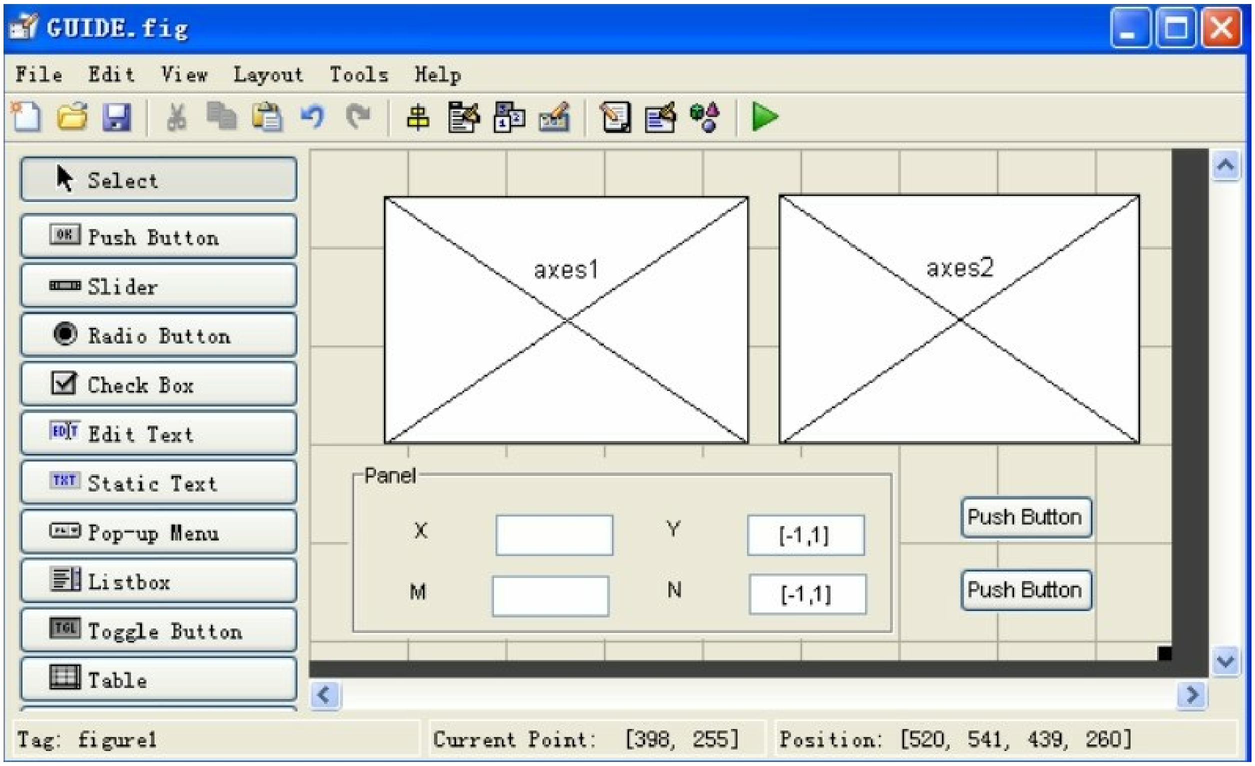


图12-22 设置右侧编辑框控件后的结果

设置“按钮”控件的名称。选择“按钮”控件，单击“Property Inspector”按钮，打开Property Inspector对话框，选择“String”选项，将其设置为“Draw”，得到的结果如图12-23所示。

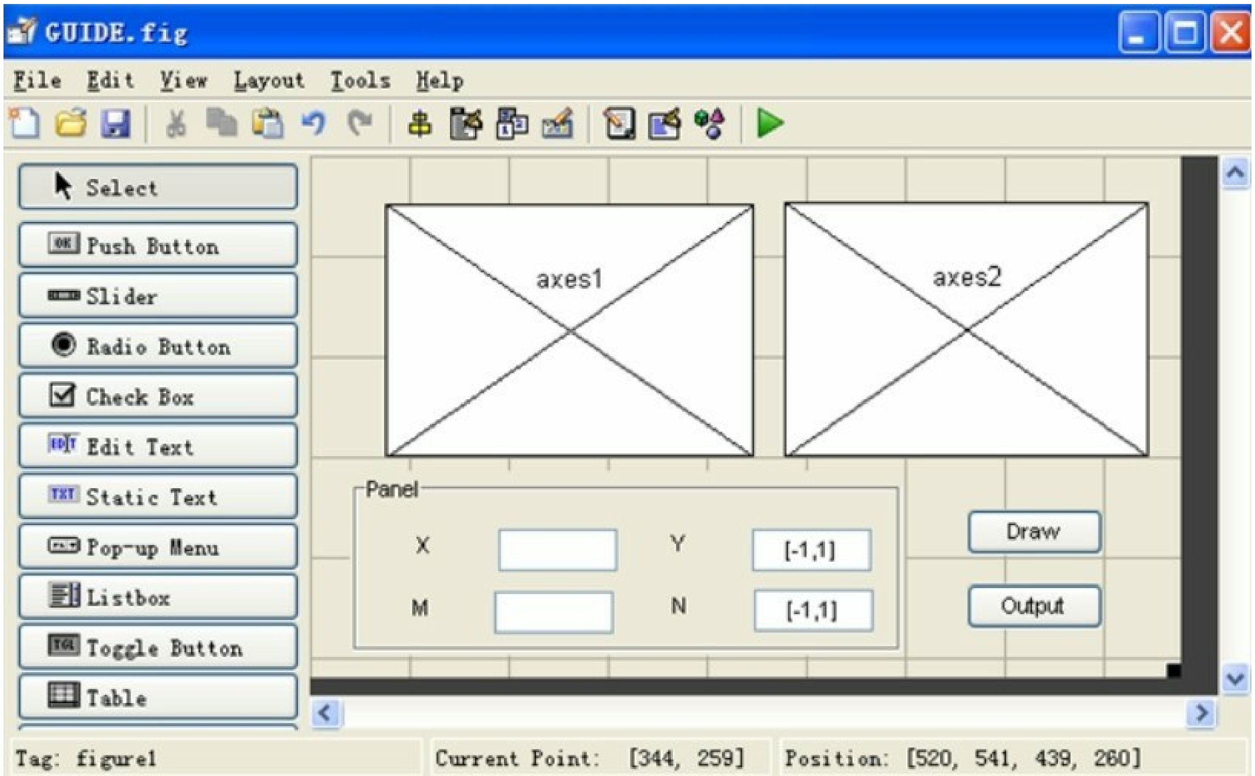


图12-23 设置“按钮”控件名称

设置“按钮”控件的属性。选择“按钮”控件，单击“Property Inspector”按钮，打开Property Inspector对话框，选择“Enable”选项，将其设置为“off”，得到的结果如图12-24所示。

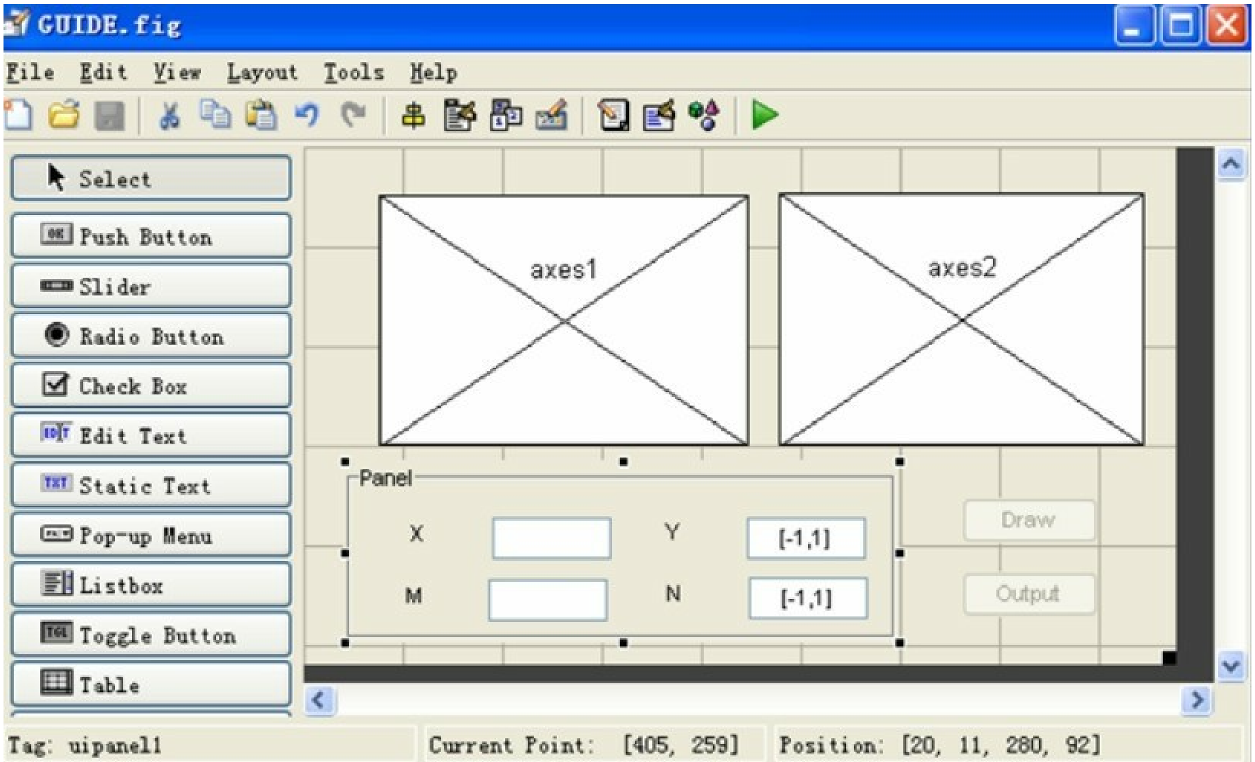


图12-24 设置按钮控件属性

查看属性设置的结果。前面步骤已经设置控件所有的相关属性，选择 GUIDE 菜单栏中的 tool-run命令，或者直接单击菜单栏中的“Run”按钮，查看属性设置的结果，如图12-25所示。

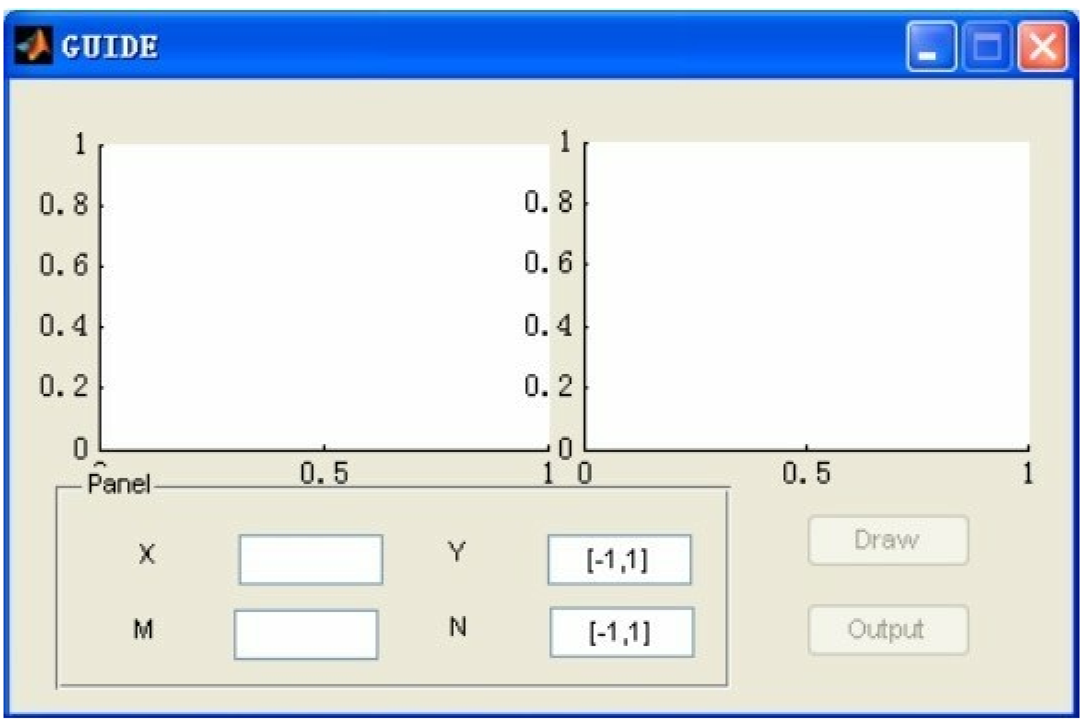


图12-25 查看属性设置的结果

如果用户是第一次运行该命令，MATLAB会提示保存该 FIG 文件，用户可以选择合适的保存路径和名称。在本例中，将其保存在用户的默认路径中，名称设置为GUItest。

12.1.5 编写相应的程序代码

延续上面小节的步骤。

设置各控件的“tag”属性。选择“Draw”按钮，单击“Property Inspector”按钮，打开Property Inspector对话框，选择“tag”选项，将其设置为“Draw”，得到的结果如图12-26所示。

重复上面的步骤，为添加的各种控件设置“Tag”属性。

打开M文件编辑器。选择GUIDE菜单栏中的View-Editor命令，或者直接单击菜单栏中的“Editor”按钮，打开M文件编辑器，如图12-27所示。

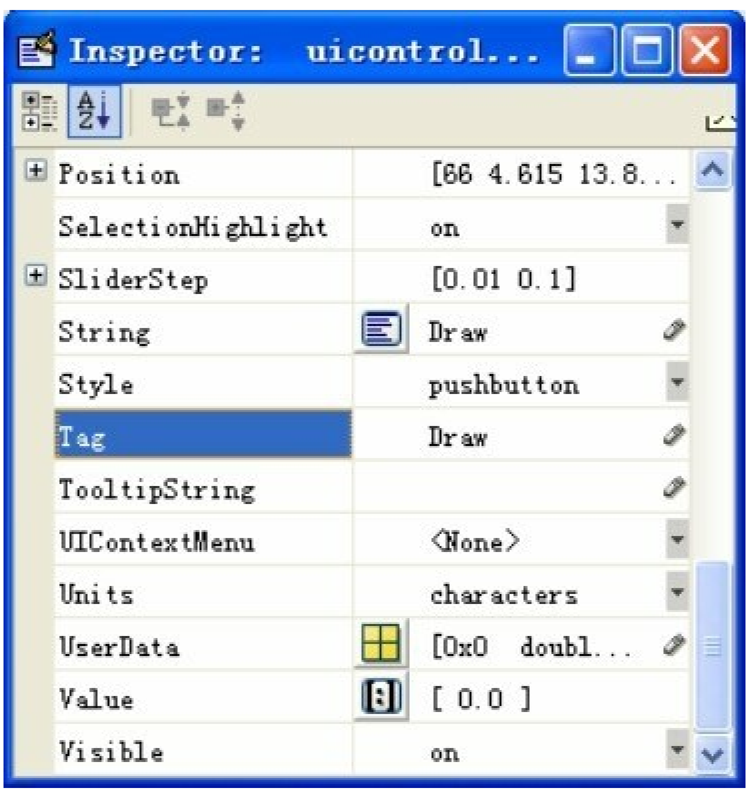


图12-26 设置控件的Tag属性

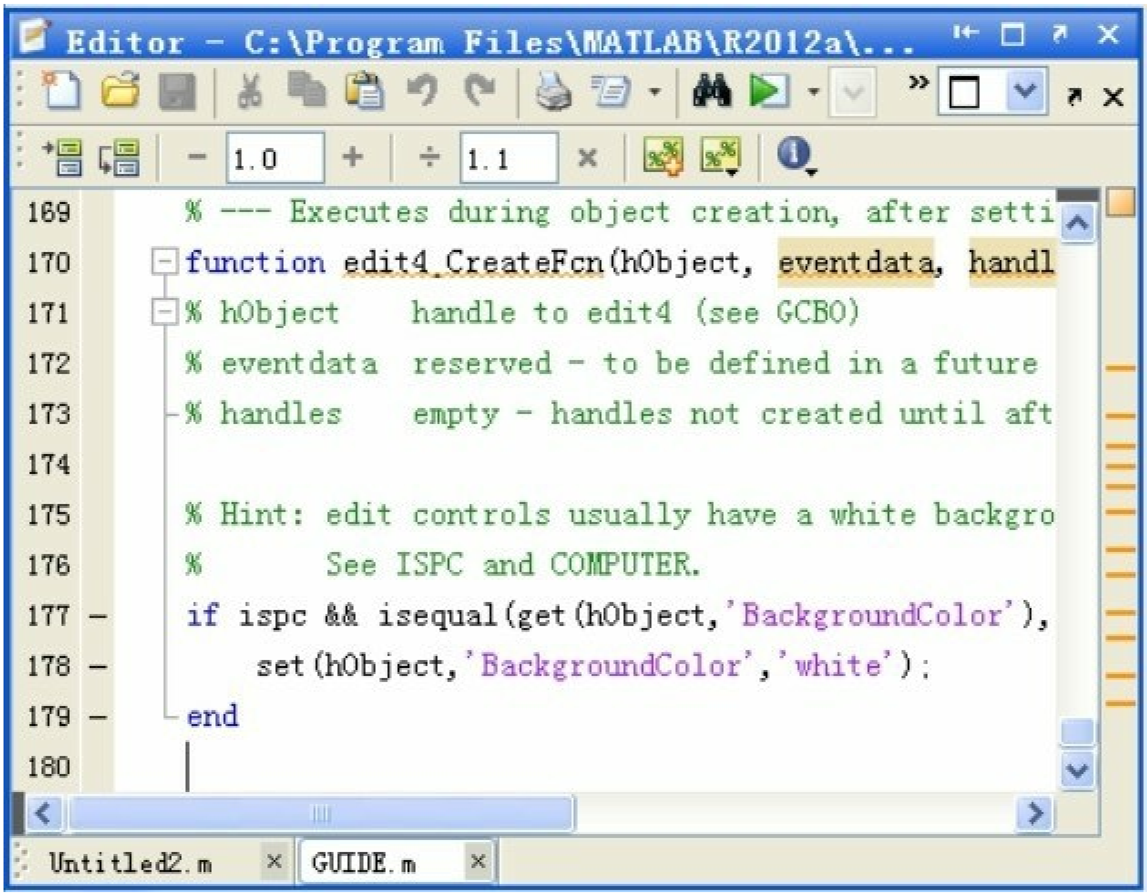


图12-27 M文件编辑器

查看 M 文件程序代码。在前面介绍过，GUIDE 会自动生成前面所有布局和属性设置的M文件，其代码程序如下。

```
function varargout = GUI(varargin)
% GUI MATLAB code for GUI.fig
%    GUI, by itself, creates anewGUI or raises the existing
%       singleton*.
%
%    H=GUI returns the handle to a newGUI or the handle to
%          the existing singleton*.
%
%    GUI('CALLBACK',hObject,eventData,handles,...) calls the local
```

% function namedCALLBACK inGUI.Mwith the given input arguments.

%

% GUI('Property','Value',...) creates a newGUI or raises the

% existing singleton*. Starting from the left, property value pairs are

% applied to theGUI beforeGUI_OpeningFcn gets called.An

% unrecognizedproperty name or invalidvaluemakes property application

% stop.All inputs are passed toGUI_OpeningFcn via varargin.

%

% *SeeGUIOptions onGUIDE'sToolsmenu.Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI

% Last Modified by GUIDE v2.5 08-Sep-2012 20:26:51

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;

gui_State= struct('gui_Name', mfilename, ...

'gui_Singleton', gui_Singleton, ...

'gui_OpeningFcn', @GUI_OpeningFcn, ...

'gui_OutputFcn', @GUI_OutputFcn, ...

'gui_LayoutFcn', [] , ...

'gui_Callback', []);

if nargin && ischar(varargin{1})

```
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%--- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata    reserved - to be defined in a futureversion ofMATLAB
% handles    structurewithhandles anduser data (seeGUIDATA)
% varargin    command line arguments toGUI (seeVARARGIN)
% Choose default command line output for GUI
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);
%--- Outputs from this function are returned to the command line.
function varargout = GUI_OutputFcn(hObject, eventdata, handles)
% varargout    cell array for returning output args (seeVARARGOUT);
% hObject    handle to figure
% eventdata    reserved - to be defined in a futureversion ofMATLAB
% handles    structurewithhandles anduser data (seeGUIDATA)
```

% Get default command line output from handles structure

varargout{1}= handles.output;

%--- Executes on button press in Draw.

function Draw_Callback(hObject, eventdata, handles)

% hObject handle toDraw (seeGCBO)

% eventdata reserved - to be defined in a futureversion ofMATLAB

% handles structurewithhandles anduser data (seeGUIDATA)

%--- Executes on button press in Output.

function Output_Callback(hObject, eventdata, handles)

% hObject handle toOutput (seeGCBO)

% eventdata reserved - to be defined in a futureversion ofMATLAB

% handles structurewithhandles anduser data (seeGUIDATA)

function Y_Callback(hObject, eventdata, handles)

% hObject handle toY (seeGCBO)

% eventdata reserved - to be defined in a futureversion ofMATLAB

% handles structurewithhandles anduser data (seeGUIDATA)

% Hints: get(hObject,'String') returns contents of Y as text

% str2double(get(hObject,'String')) returns contents ofY as a double

%--- Executes during object creation, after setting all properties.

function Y_CreateFcn(hObject, eventdata, handles)

% hObject handle toY (seeGCBO)

% eventdata reserved - to be defined in a futureversion ofMATLAB

% handles empty -handles not created until after allCreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC andCOMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'),

get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function X_Callback(hObject, eventdata, handles)

% hObject handle toX (seeGCBO)

% eventdata reserved - to be defined in a futureversion ofMATLAB

% handles structurewithhandles anduser data (seeGUIDATA)

% Hints: get(hObject,'String') returns contents of X as text

% str2double(get(hObject,'String')) returns contents ofX as a double

set(handlee.Draw,'Enable','On');

function is available

guidata(hObject,handles)

%--- Executes during object creation, after setting all properties.

function X_CreateFcn(hObject, eventdata, handles)

% hObject handle toX (seeGCBO)

% eventdata reserved - to be defined in a futureversion ofMATLAB

% handles empty -handles not created until after allCreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC andCOMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function N_Callback(hObject, eventdata, handles)

% hObject handle toN (seeGCBO)

% eventdata reserved - to be defined in a futureversion ofMATLAB

```
% handles    structurewithhandles anduser data (seeGUIDATA)
% Hints: get(hObject,'String') returns contents of N as text
%        str2double(get(hObject,'String')) returns contents ofN as a double
%--- Executes during object creation, after setting all properties.
function N_CreateFcn(hObject, eventdata, handles)
% hObject    handle toN (seeGCBO)
% eventdata    reserved - to be defined in a futureversion ofMATLAB
% handles    empty -handles not created until after allCreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC andCOMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function M_Callback(hObject, eventdata, handles)
% hObject    handle toM (seeGCBO)
% eventdata    reserved - to be defined in a futureversion ofMATLAB
% handles    structurewithhandles anduser data (seeGUIDATA)
% Hints: get(hObject,'String') returns contents of M as text
%        str2double(get(hObject,'String')) returns contents ofM as a double
%--- Executes during object creation, after setting all properties.
function M_CreateFcn(hObject, eventdata, handles)
% hObject    handle toM (seeGCBO)
% eventdata    reserved - to be defined in a futureversion ofMATLAB
% handles    empty -handles not created until after allCreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC andCOMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

上面的程序代码代表了使用GUIDE创建GUI对象的典型代码结构，下面详细介绍其中各组成部分的结构。

函数名称。function varargout-vectguf，MATLAB 会根据用户保存的名称定义该GUI 对象的主函数名称。在本实例中使用的名称为GUIDE。

函数的注释文字。从“% hObject handle to edit4 (see GCBO)”到“%See ISPC and COMPUTER”都是MATLAB自动生成的代码。

GUI 对象的初始化代码。从“% Begin initialization code - DO NOT EDIT”到“% End initialization code - DO NOT EDIT”代码部分，是MATLAB 自动生成的GUI 对象初始化程序代码，这部分代码用户不能编辑（DO NOT EDIT）。

函数vectgui_OpeningFcn的程序代码。该函数也是MATLAB自动设置的函数，该部分程序代码在GUI可见之前执行，也就是在打开GUI对象之前执行的程序代码。

函数vectgui_OutputFcn的程序代码。该函数自动设置空间的属性。对于用户在GUIDE中添加的所有空间，MATLAB都会设置对应的创建函数。

控件的回调函数。这部分函数是编辑 GUI 的 M 文件主要部分，主要功能就是编写互动事件的程序。例如，函数“Draw_Callback”的功能就是编写单击“Draw”按钮所相应的事件和行为。因此，回调函数是实现

GUI互动功能的主要部分，也就是用户编写代码的主要部分。

编写“X_Callback”控件的回调函数。为了方便用户编写回调函数，将M文件编辑器回嵌到 MATLAB 的命令窗口中，然后选择“X_Callback”选项，MATLAB 会自动跳到程序代码的相应位置，如图12-28所示。

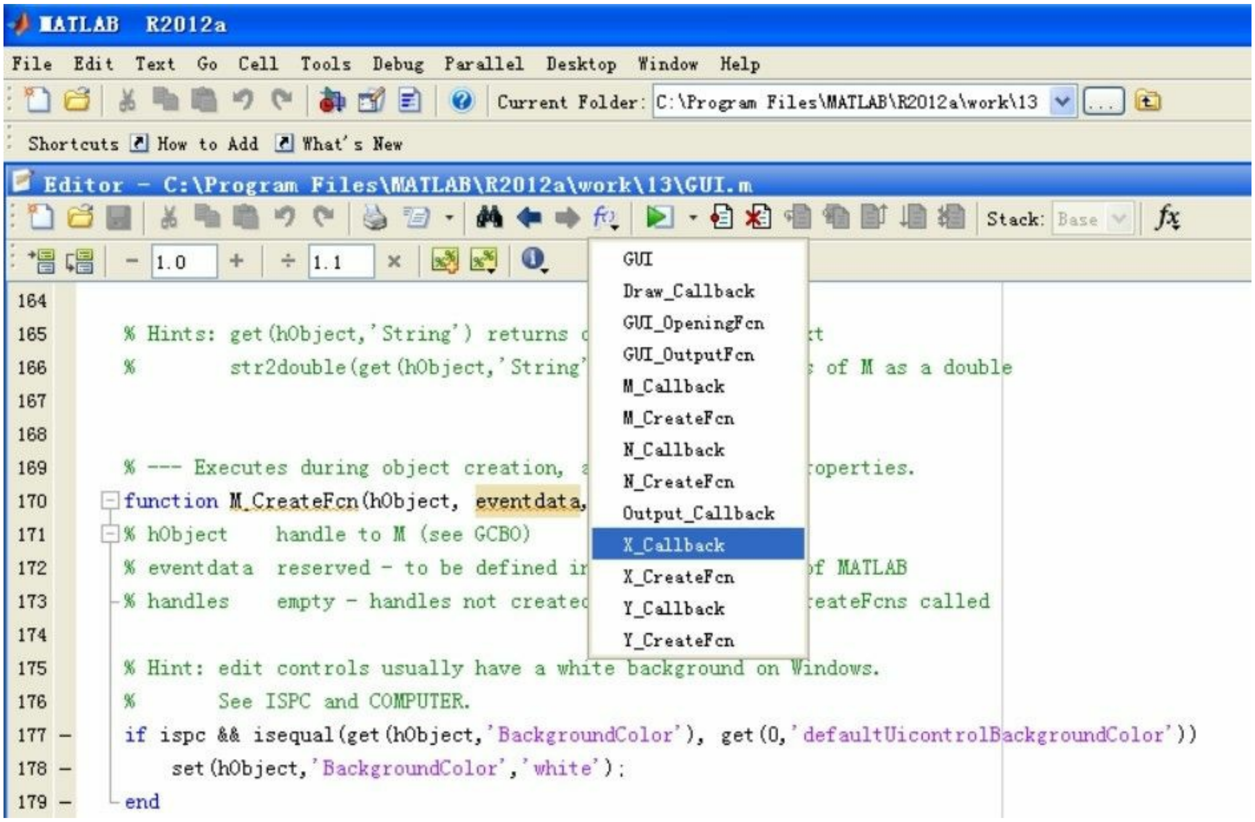


图12-28 选择对应的程序代码

在MATLAB转到的位置上，编写对应的程序代码，详细代码如下。

```
function X_Callback(hObject, eventdata, handles)
% hObject    handle toX (seeGCBO)
% eventdata    reserved - to be defined in a futureversion ofMATLAB
% handles    structurewithhandles anduser data (seeGUIDATA)
% Hints: get(hObject,'String') returns contents of X as text
% str2double(get(hObject,'String')) returns contents of X as a double
```

```
set(handlee.Draw,'Enable','On');
function is available
guidata(hObject,handles)
```

在上面的程序代码中，只有后面两行代码是用户填写的，注释文字是MATLAB自动生成的，用户可以不用编写或者修改。上面两行代码的功能是，当用户在编辑框中输入相应的表达式后，将 Draw 按钮功能设置为可用，然后将编辑框中输入的数值传递到相应的变量中。

对应GUI对象中左侧的两个编辑框，用户可以编写对应的回调函数，其具体的程序代码和上面控件程序完全不同。

编写M的回调函数。在MATLAB命令窗口中，选择“M_Callback”选项，编写对应的程序代码，编写后的代码如下。

```
function M_Callback(hObject, eventdata, handles)
% hObject    handle toM (seeGCBO)
% eventdata    reserved - to be defined in a futureversion ofMATLAB
% handles    structurewithhandles anduser data (seeGUIDATA)
% Hints: get(hObject,'String') returns contents of M as text
% str2double(get(hObject,'String')) returns contents of M as a double
as a double
guidata(hObject,handles)
```

上面的程序代码功能十分简单，就是将用户在“M”控件中输入的数值传递给对应的程序变量。对于其他的编辑框控件，用户可以添加相同的程序代码。

12.1.6 GUIDE创建GUI的注意事项

前面已经分步骤详细介绍了如何创建一个简单的GUI对象，在本小节中，将介绍使用GUIDE来创建GUI的一些主要注意事项。为了能够更

加方便地介绍这些注意事项，还是沿用上面小节使用的GUI对象为例。

首先，在GUIDE中可以使用多种工具为多个控件对象进行精细排列，其中“排列对象”对话框是比较常用的工具。使用GUIDE打开上面小节中的GUI对象，然后同时选取两个“按钮”控件，选择菜单栏中的Tool-Align Objects 命令，打开Align Objects对话框，如图12-29所示。

在上面的对话框中，可以选取多个排列方式，同时为每个排列方式设置具体的间距，单位是pixels（像素），当选定了对应的排列方式和间距后，单击对话框中的“Apply”按钮，就可以查看修改的排列结果。

如果添加的是单个控件，而不是空间组，那么上面的对话框就很难有效果。如果使用鼠标来移动控件的单位移动量过大，可以选中控件后，使用方向键来移动控件，对控件的位置进行微调。

除了上面介绍的Align Objects对话框外，MATLAB 还提供Grid and Rulers对话框，可以在该对话框中设置网格和标线的属性。选择菜单栏中“tools”下拉菜单命令，打开Grid and Rulers对话框，如图12-30所示。

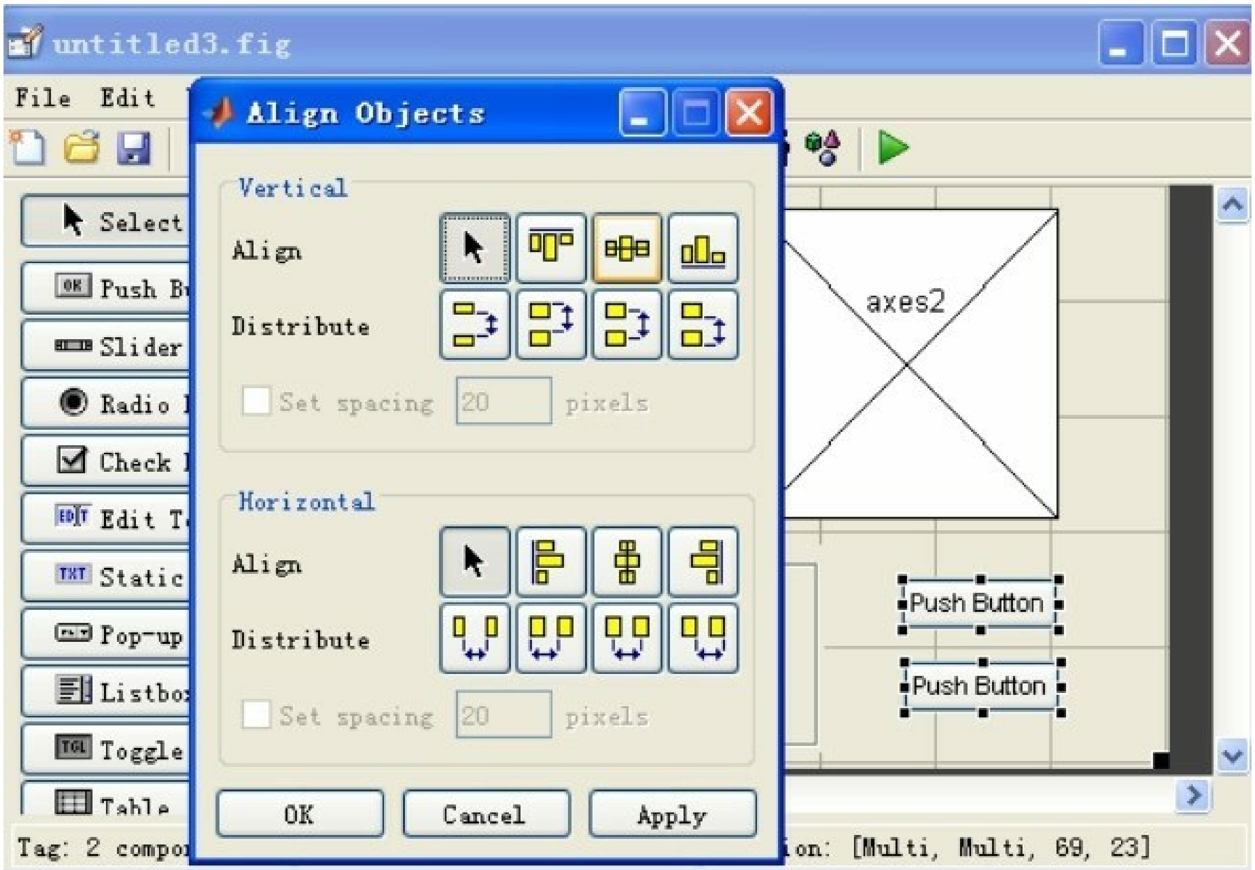


图12-29 排列对象对话框

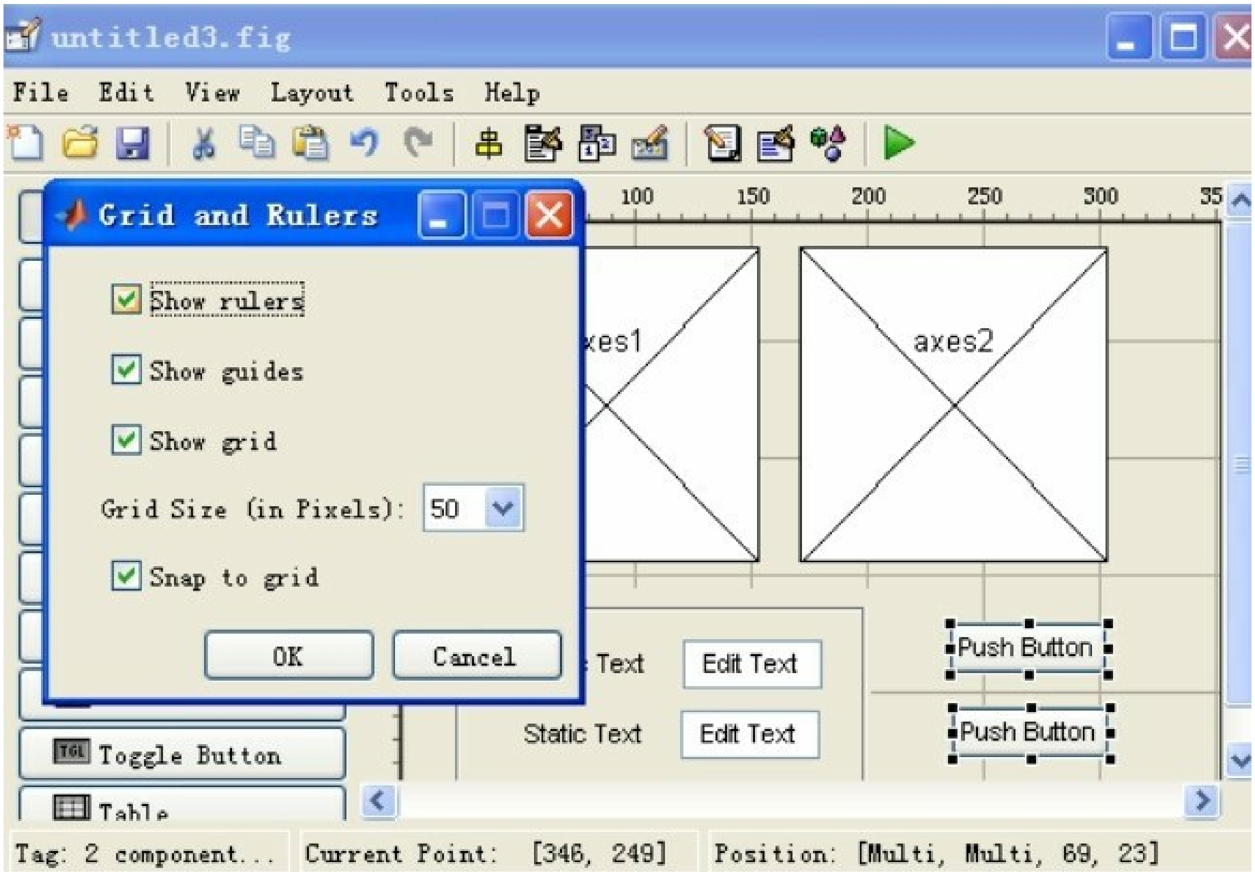


图12-30 网格和标线对话框

在上面的对话框中，可以控制标线、网格的属性，设置网格间距等。在默认情况下，网格线的间隔是50 像素，可以根据需要自行设置间距数值，MATLAB接受的间距数值范围是10～200 像素。当选中上面的对话框中的“Show rulers”选项后，将会在GUI 界面周围出现标尺。

在MATLAB的GUIDE中，除了提供了可以精细排列控件的工具之外，还提供了查看各种对象的对象浏览器对话框。选择GUIDE 菜单栏中的View-Object Browser 命令，打开Object Browser对话框，如图12-31所示。

在上面的对象浏览器中，用户可以查看各个图形对象的名称和继承关系，当GUI对象中包含多个空间时，在该对象浏览器中可以很方便地实现对象管理。

在本小节的最后，将介绍设定 GUI 组态的对话框工具。选择

GUIDE 菜单栏中的tools-GUI Options 命令，打开GUI Options 对话框，在该对话框中可以设置GUI 初始组态的属性，如图12-32所示。

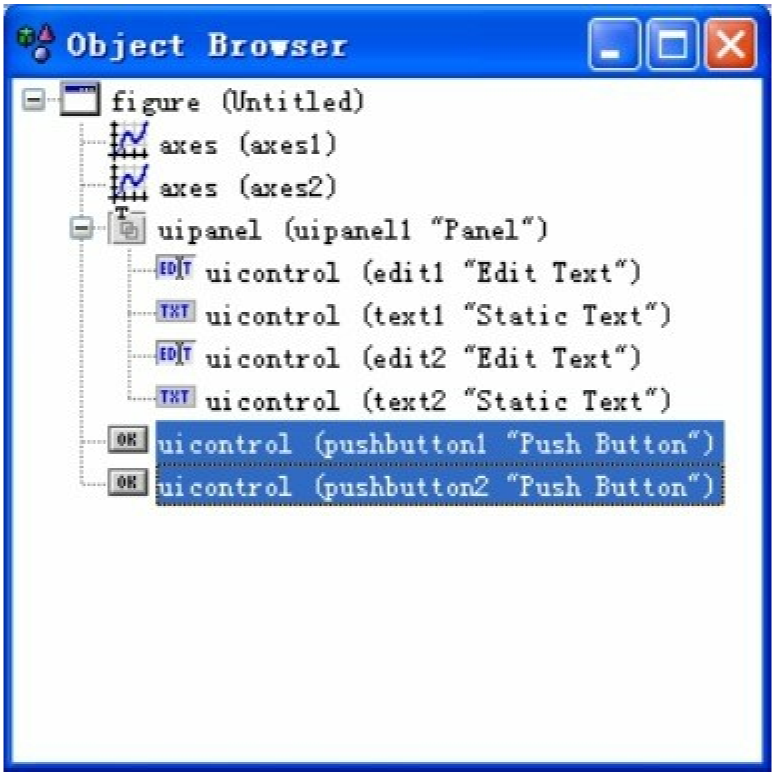


图12-31 对象浏览器

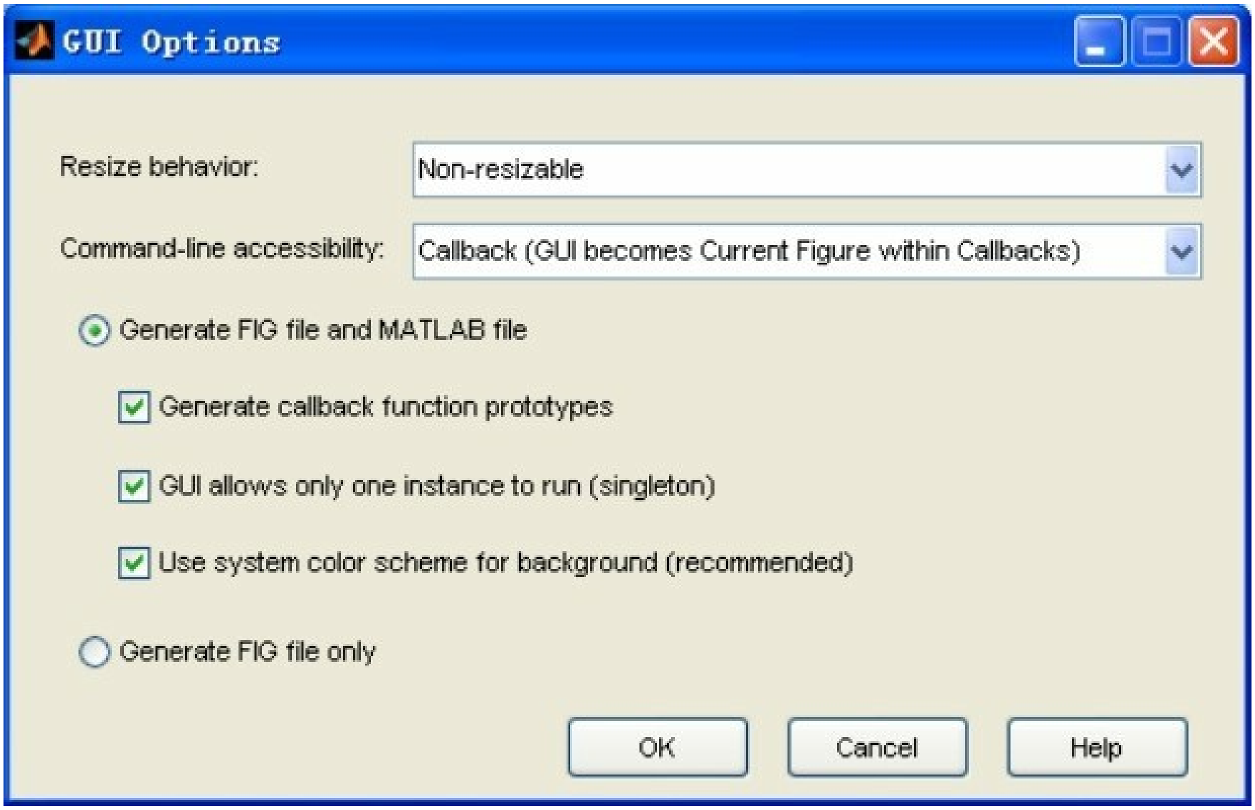


图12-32 GUI应用程序的选项对话框

在创建GUI对象的时候，上面对话框中的各个属性都是十分重要的。为了帮助用户更好地理解GUI的各种属性，下面详细介绍对话框各个选项的具体含义。

12.1.7 定制标准菜单

在GUI控件对象中，界面菜单是一个重要组成部分。在句柄图形对象结构中，Uimenu对象的结构体系以Figure图形窗口为父对象，和Axes坐标轴、Unicontrol界面控件为平等级别的组件。

在MATLAB中，可以根据需要在GUI对象中创建标准菜单，自行设置菜单或者创建现场菜单等。同时，可以设置菜单控件的各种属性，例如添加快捷键、设置对应的回调函数等。因此，在 GUI 对象中，可以设置菜单控件来完成多种功能。对于比较简单的 GUI对象，可以根据需要定制MATLAB图形窗口的标准菜单，设置不同的菜单属性。下面介

绍一个简单的实例，来说明如何定制标准菜单。

【例12-2】创建一个简单的GUI对象，根据需要定制标准菜单。

解：创建一个默认的图形窗口。在MATLAB的命令窗口中输入下面的命令代码。

>> Handle_figure=figure;

查看图形结构。输入上面的代码后，按 Enter 键，得到默认的图形窗口，如图 12-33 所示。



图12-33 创建默认图形窗口

在默认情况下，MATLAB 的图形窗口有一个顶层菜单条（Top-Level Menu），在MATLAB 中，该菜单条包括 8 个菜单项目：File、Edit、View、Insert、Tools、Desktop、Window和Help。当用户单击每一菜单项时，都会产生对应的下拉菜单。

查看Menu属性列表。在命令窗口中输入命令set(Handle_figure,'MenuBar')，得到的结果如下。

>> set(Handle_figure,'MenuBar')

[none |{figure}]

从上面的结果中可以看出，标准菜单“MenuBar”有两个属性[none | {figure}]。当对象的属性选取none时，表示图形窗口不显示标准菜单以及对应的工具条。当对象的属性选取figure属性时，图形窗口将显示标准菜单，这是图形窗口菜单的默认属性。

隐藏标准菜单。在有些程序项目中，为了不让用户对程序对象进行操作，可以隐藏菜单。在命令窗口中输入下面的代码。

```
>> set(Handle_figure,'MenuBar','none')
```

查看图形结构。输入代码后，按Enter键，得到的结果如图12-34所示。



图12-34 隐藏菜单选项后的图形窗口

恢复标准菜单。在命令窗口中输入下面的代码。

```
>> set(Handle_figure,'MenuBar','Figure');
```

查看图形结构。输入代码后，按Enter键，得到的结果如12-33所示。

12.2 神经网络GUI

在Simulink软件的神经网络工具箱中，有多种神经网络工具箱GUI，本节除了介绍常规神经网络GUI，还介绍常用的3种专业神经网络GUI。

12.2.1 常规神经网络GUI

在MATLAB的Command Window窗口输入nntool可以打开神经网络GUI 工具窗口，具体如图12-35所示。

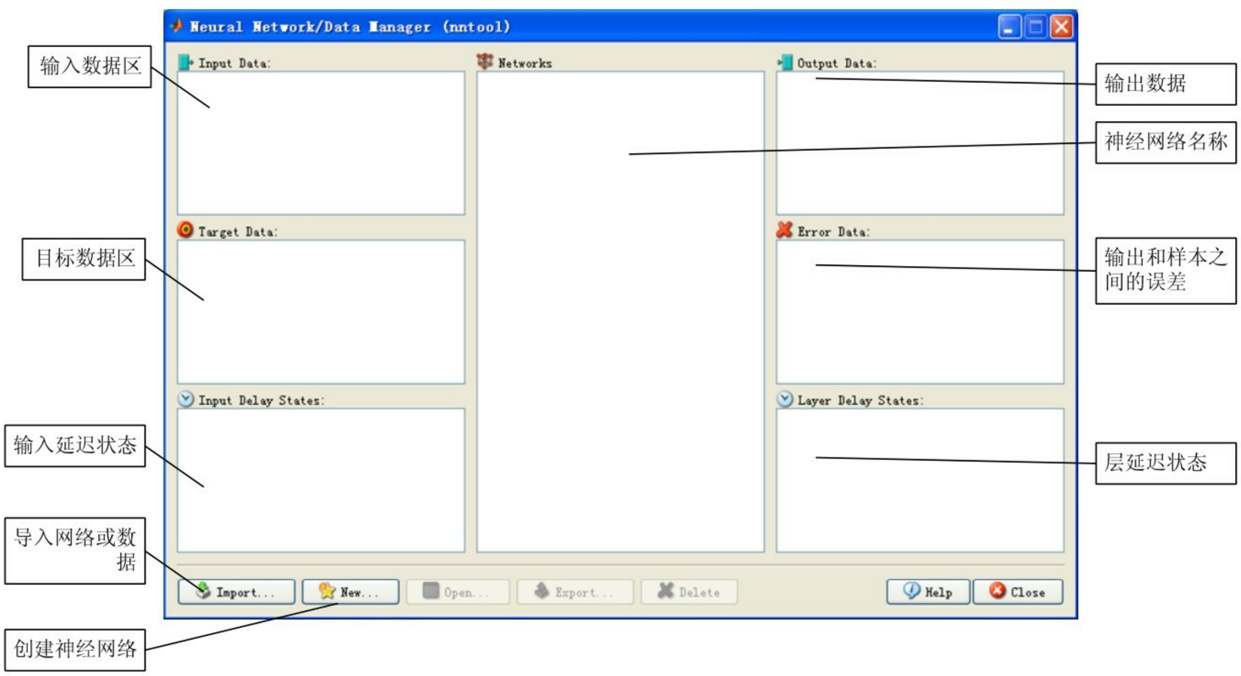


图12-35 神经网络GUI工具窗口

从图12-35可以看到，使用这个GUI工具可以非常方便地建立一个神经网络，并对网络进行训练、仿真，还可以很方便地管理神经网络的输入和输出数据。

单击图12-35中的按钮可以建立一个新的神经网络设置窗口，如图12-36所示。这个窗口包含了建立一个神经网络需要的各种操作。

在图 12-36 中，可以设置创建神经网络的名称、类型、传递函数、层数、每层神经元数等参数。单击图中按钮，可以得到图12-37所示的神经网络数据设置窗口。

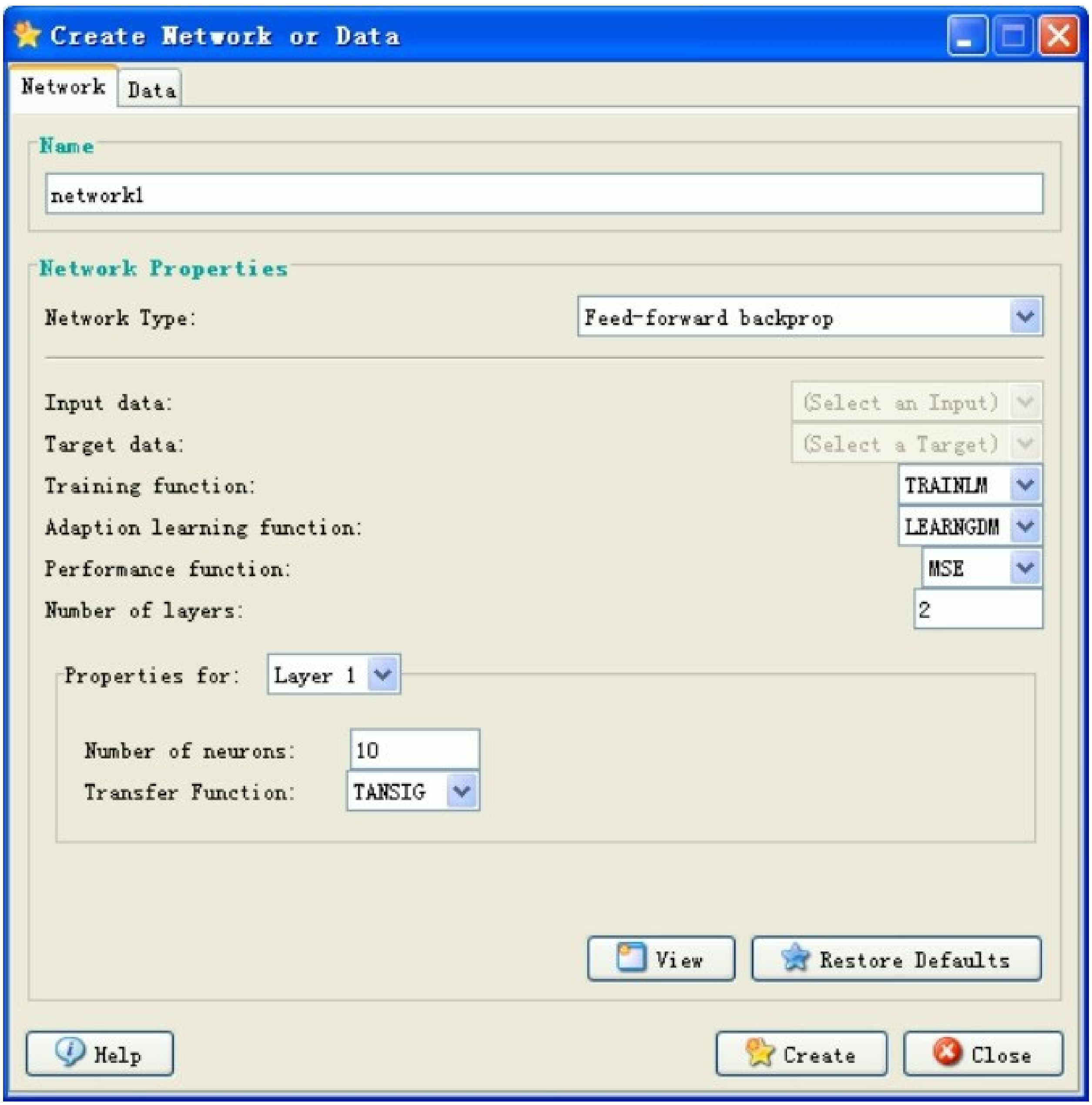


图12-36 新的神经网络设置窗口

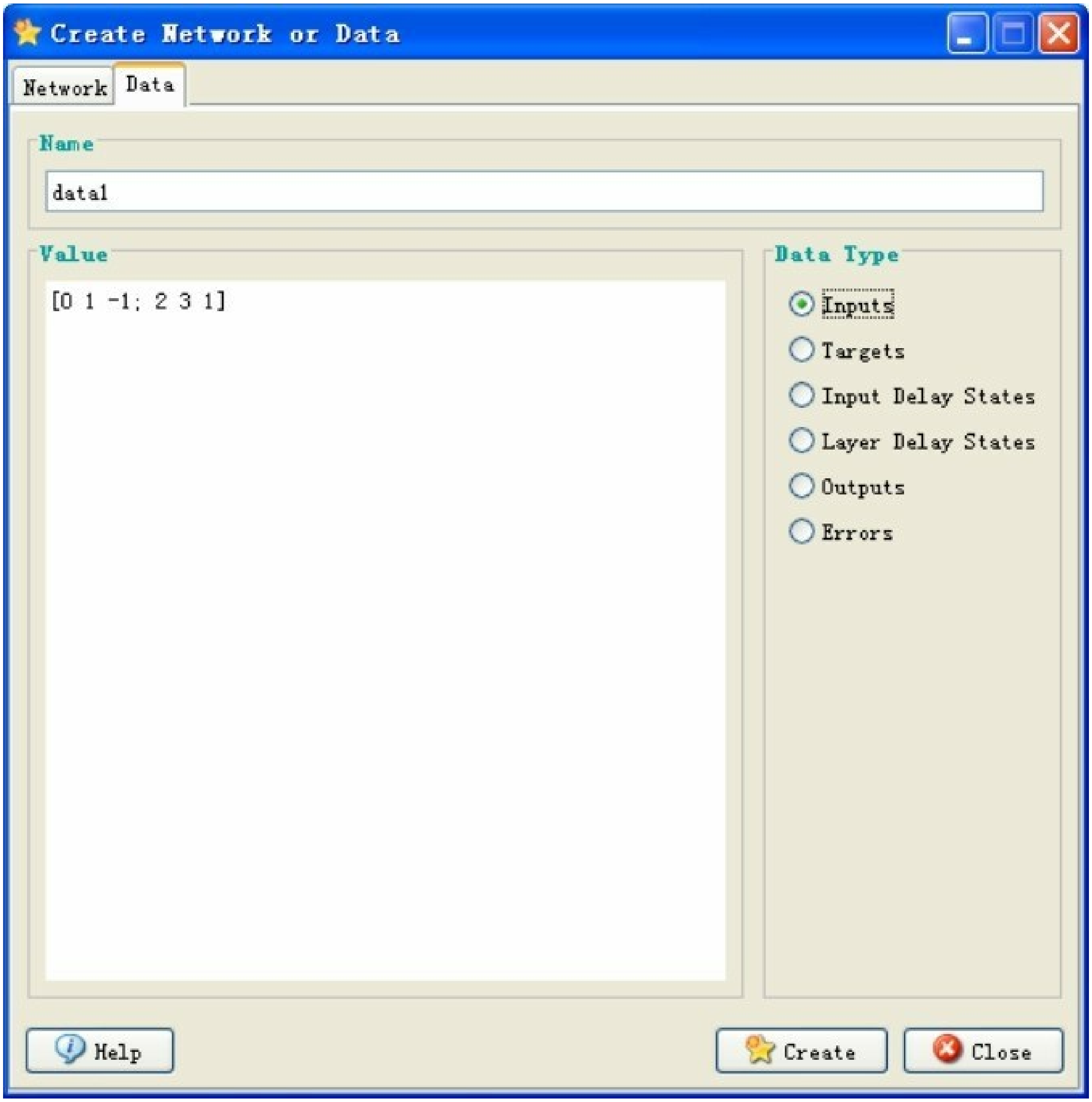
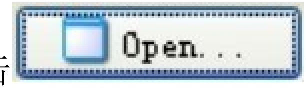


图12-37 神经网络数据设置窗口

在图12-37 中，可以设置神经网络的Inputs（输入数据）、Targets（目标数据）、Input Delay States（输入延迟状态）、Layer Delay States（层延迟状态）等参数。

创建神经网络network1后，神经网络GUI工具窗口变为图12-38所示窗口。

选择 Networks 选项下的Network1，单击

按钮，建立神经网络。得到建立的神经网络结构图窗口如图12-39所示。

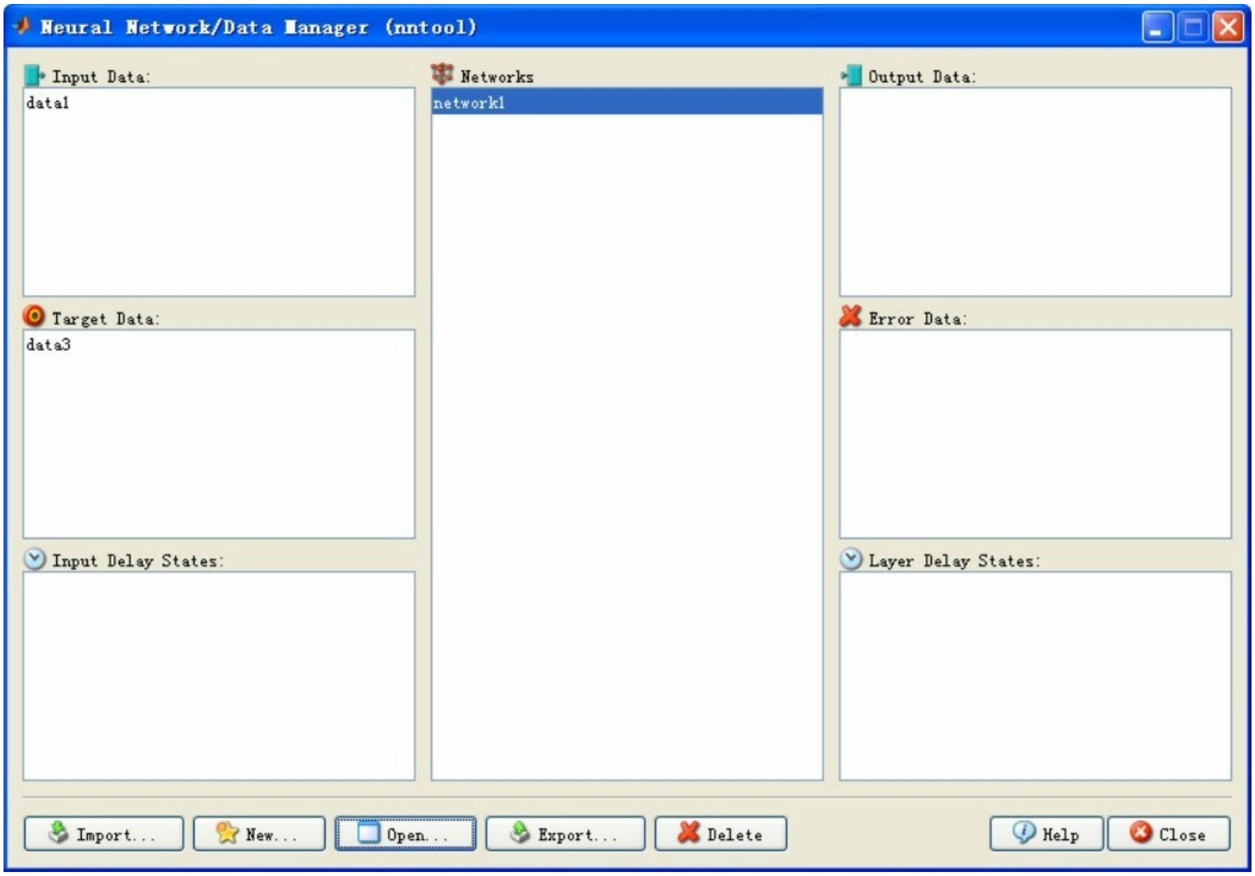


图12-38 创建神经网络network1后的GUI工具窗口

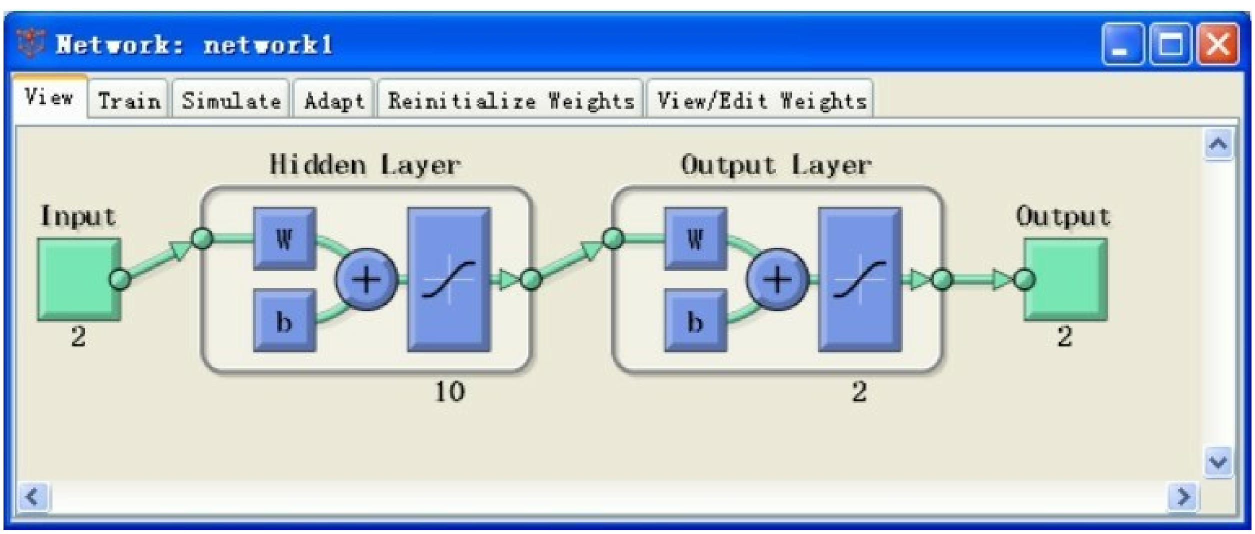


图12-39 建立的神经网络结构图窗口

单击图12-39中的“Train”选项，得到神经网络训练参数设置窗口，其中神经网络训练数据和训练结果参数设置如图 12-40 所示，神经网络训练的学习速率、训练步进等参数设置如图12-41所示。

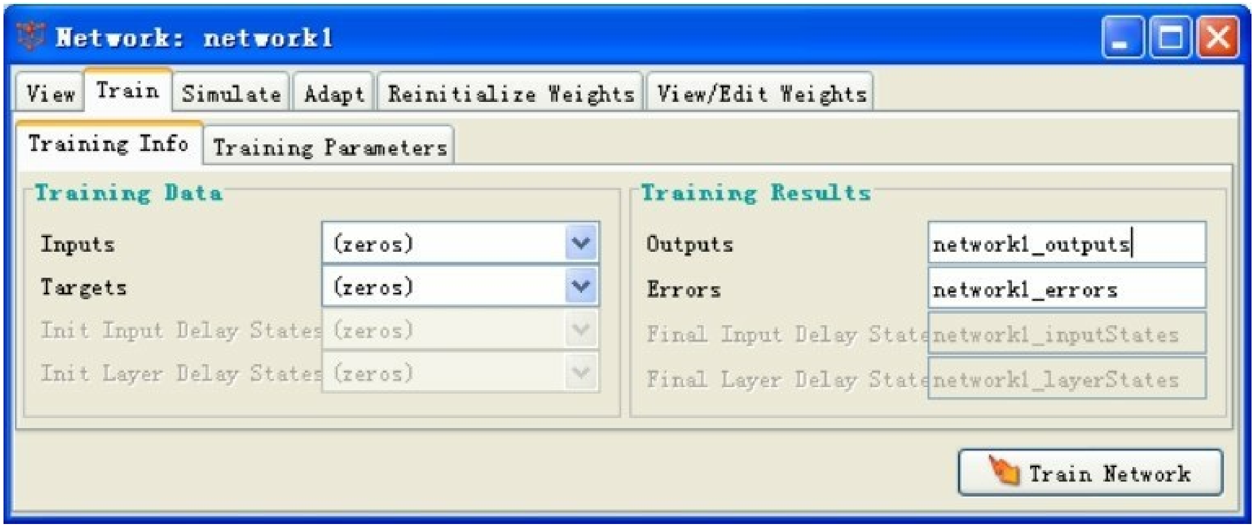


图12-40 神经网络训练数据和训练结果参数设置窗口

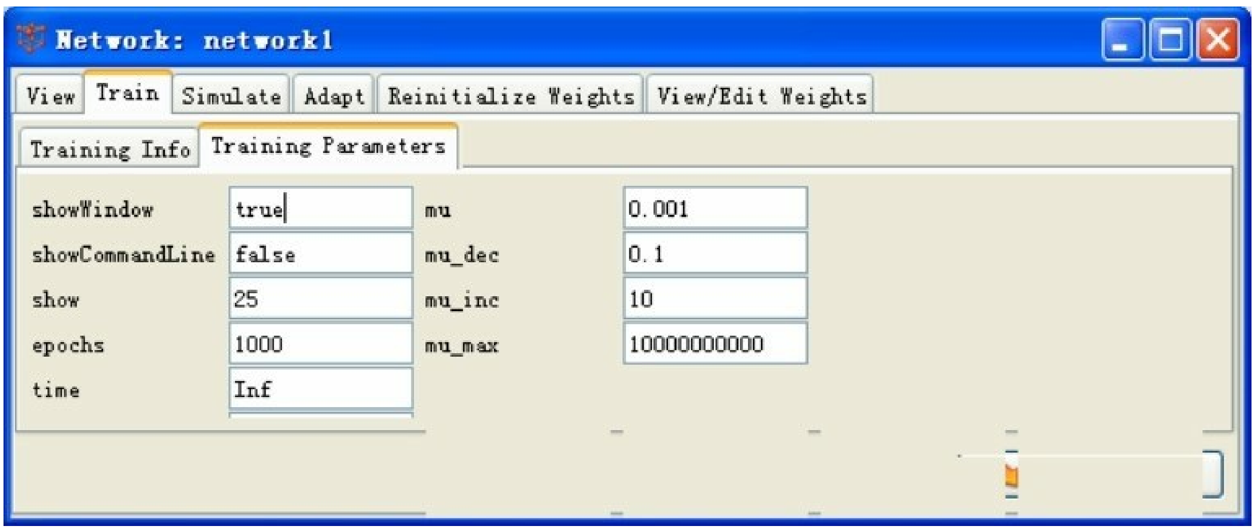


图12-41 神经网络训练的学习速率、训练步进等参数设置

神经网络的学习速率、训练步进等参数保持默认设置，输入数据和目标数据分别设置为data1和data3，如图12-42所示。

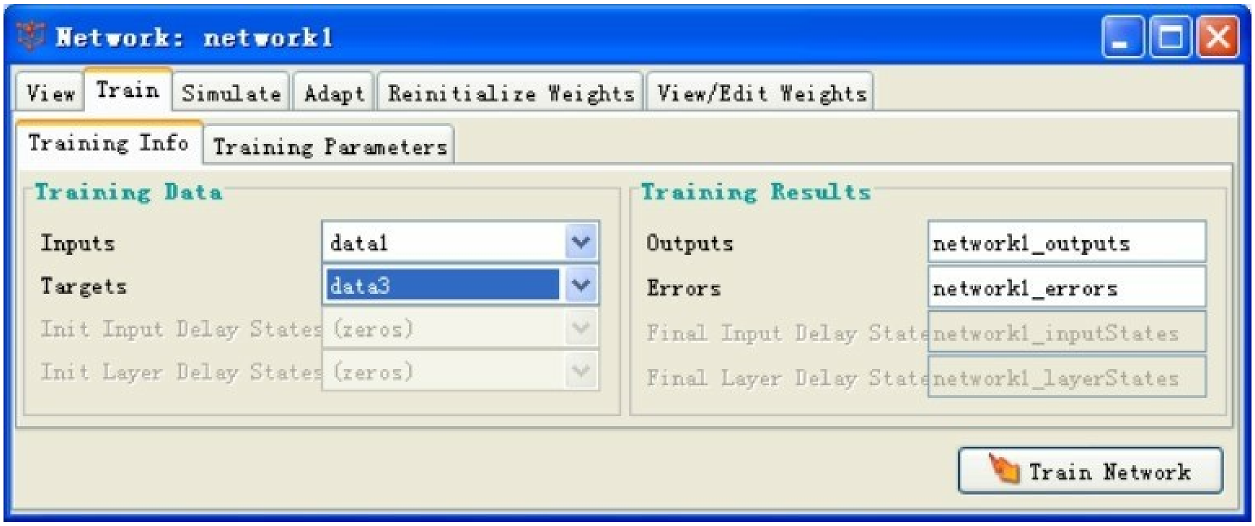


图12-42 神经网络训练参数设置

在建立的神经网络结构图窗口中选择“Adapt”选项，可以打开神经网络自适应参数设置对话框，如图12-43所示。

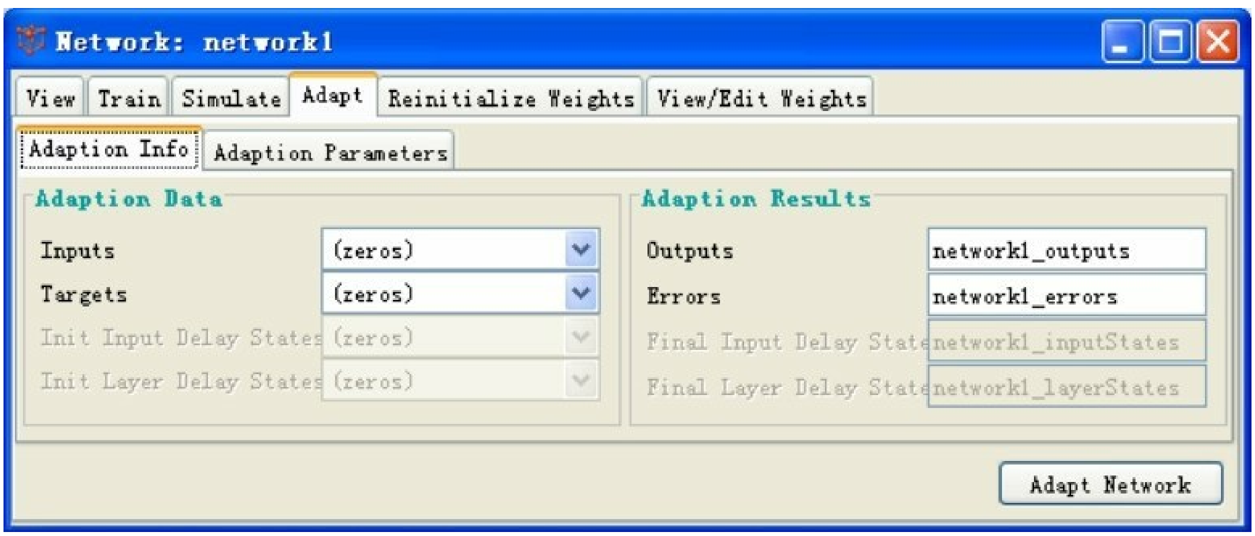


图12-43 神经网络自适应参数设置

该窗口主要用来设置网络的自适应训练参数。自适应训练是一种变步长训练过程，可以自动地改变训练步长，从而提高网络的训练速度，缩减训练时间。

在建立的神经网络结构图窗口中选择“Reinitialize Weights”选项，可以选择重新设置网络的权值和阈值，设置窗口如图12-44所示。

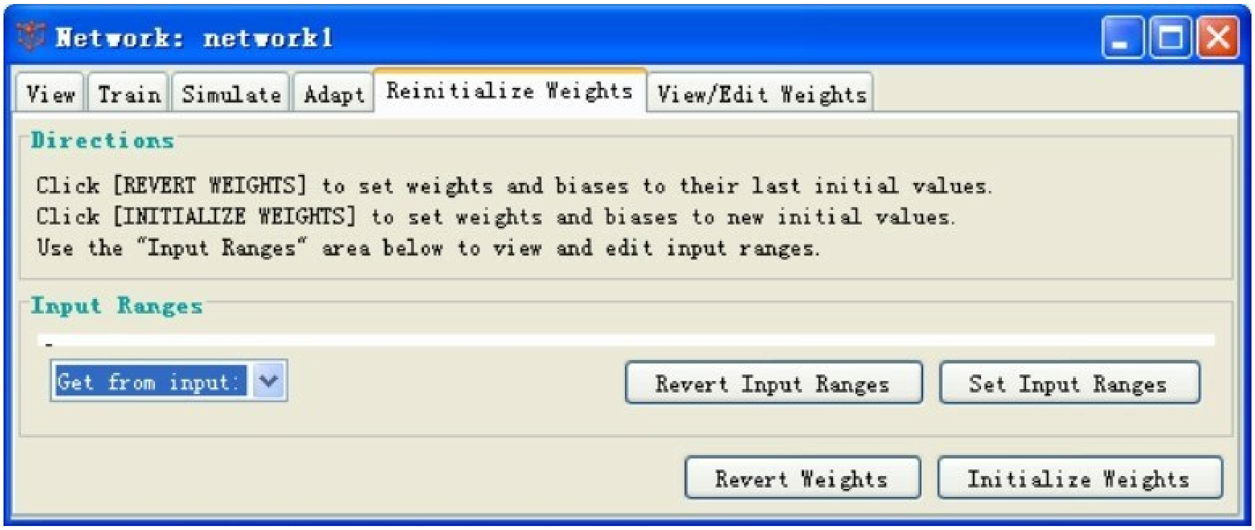


图12-44 重新设置网络的权值和阈值窗口

在建立的神经网络结构图窗口中选择“View/Edit Weights”选项，可以选择查看和编辑神经网络的权值和阈值，如图12-45所示。

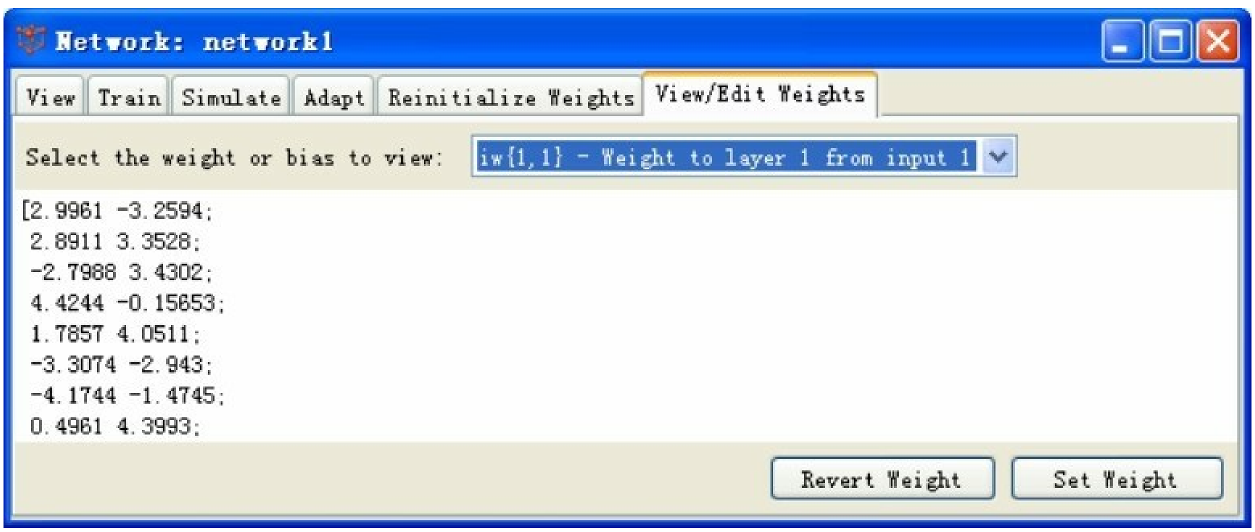
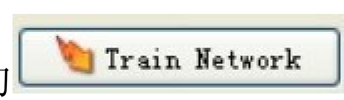


图12-45 查看和编辑网络的权值和阈值窗口

设置完网络参数后，可以单击图12-40中的

按钮训练神经网络。得到神经网络训练窗口如图12-46所示。

神经网络训练的误差曲线如图12-47所示。

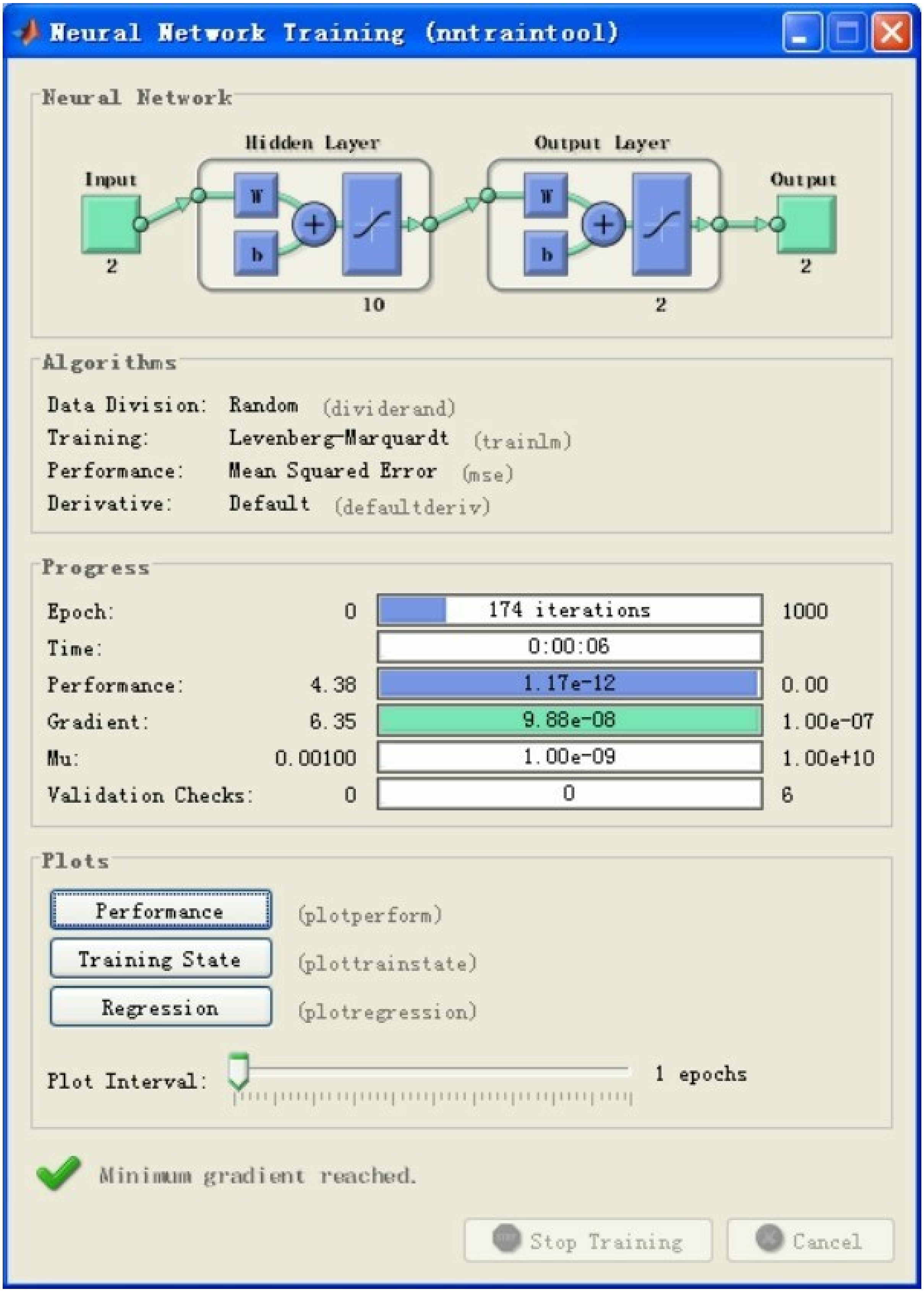

Neural Network Training (nntraintool)
Neural Network
Hidden Layer
Output Layer
Input
Output
2
10
2
2
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)
Progress
Epoch: 0 174 iterations 1000
Time: 0:00:06
Performance: 4.38 1.17e-12 0.00
Gradient: 6.35 9.88e-08 1.00e-07
Mu: 0.00100 1.00e-09 1.00e+10
Validation Checks: 0 0 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)
Plot Interval: 1 epochs
Minimum gradient reached.
Stop Training
Cancel

图12-46 神经网络训练窗口

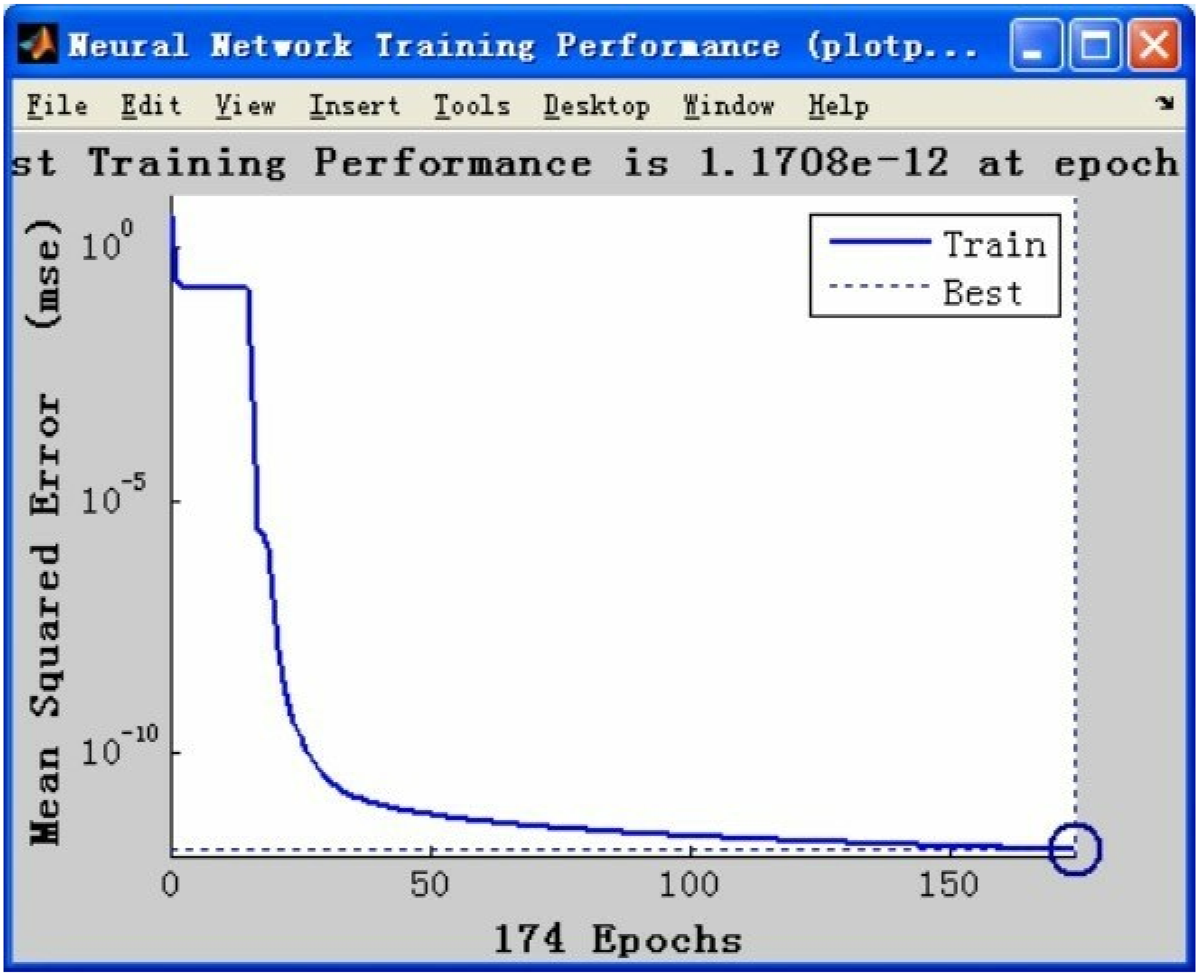


图12-47 神经网络训练的误差变化曲线

完成神经网络的训练后，在建立的神经网络结构图窗口中选择“Simulate”选项，可以设置神经网络的仿真参数，这里选择data1作为神经网络的仿真输入数据，如图12-48所示。

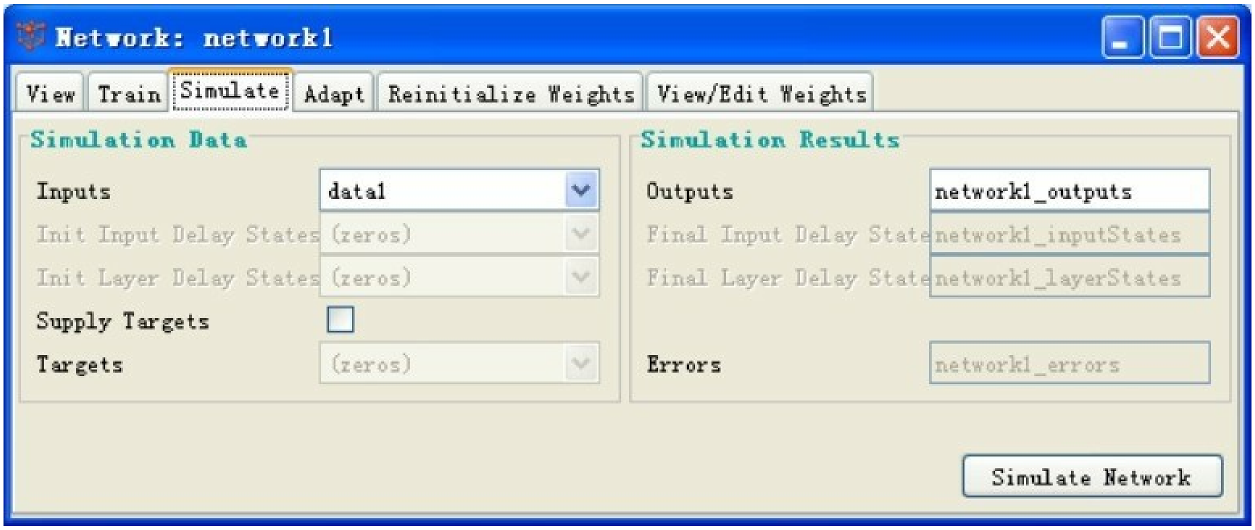


图12-48 设置仿真参数

单击图12-48中的

按钮，可以得到图12-49所示的提示。这表明仿真操作已经完成，在Network/Data Manager窗口可以查看仿真结果。

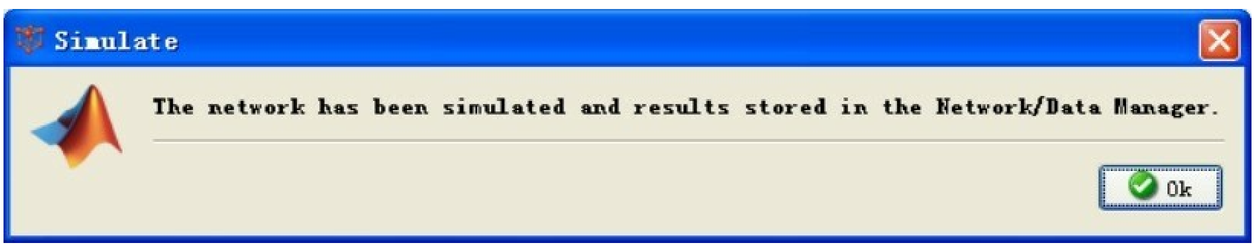


图12-49 神经网络完成仿真的提示

仿真完成后的Network/Data Manager窗口如图12-50 所示。

双击图12-50中的network1_outputs，可以得到图12-51所示的网络仿真输出结果。双击data1可以得到网络的目标值，如图12-52所示。

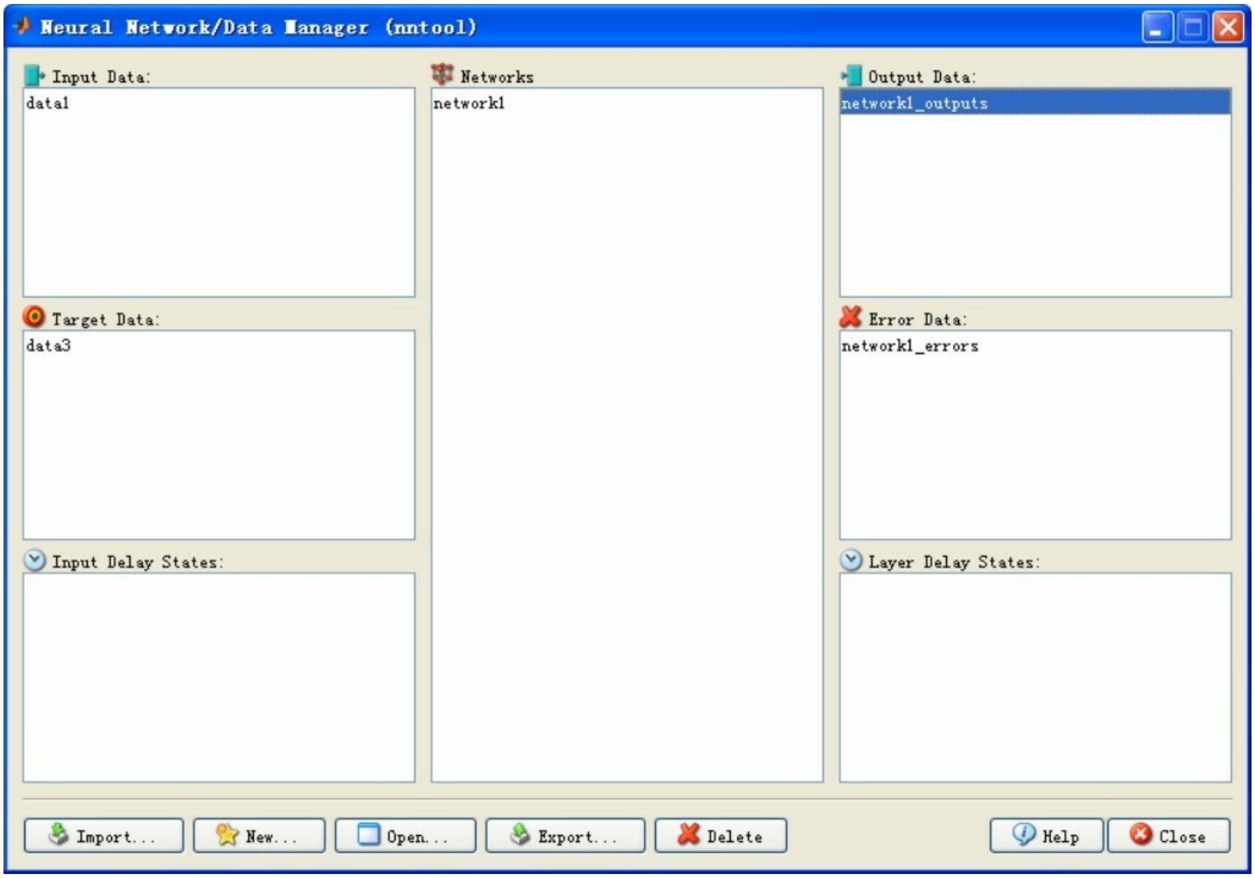


图12-50 仿真完成后的Network/Data Manager窗口

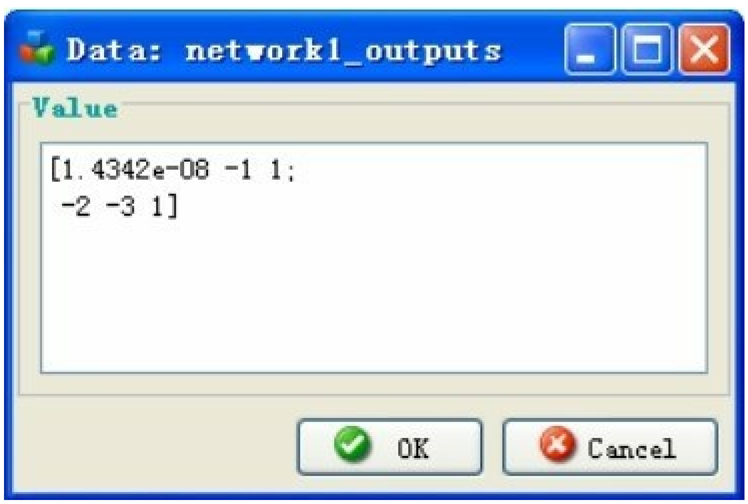


图12-51 网络仿真输出结果

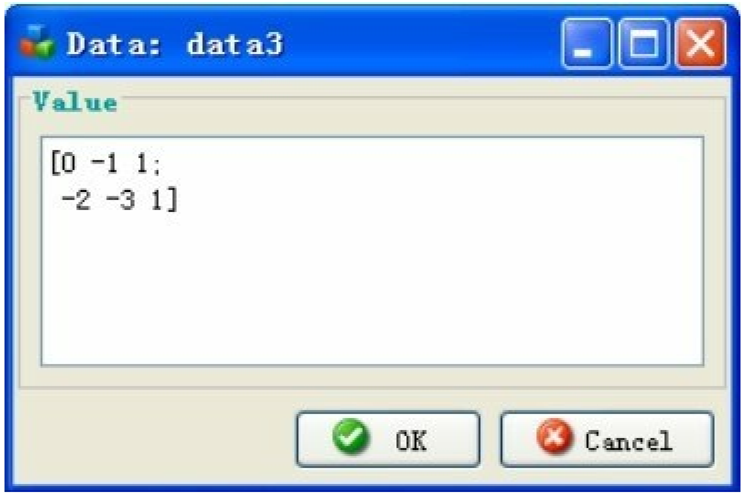


图12-52 网络仿真目标值

由图 12-51 和图 12-52 可以知道，网络输出值与网络仿真目标值基本一致，说明通过上述方法建立神经网络是可行的。

12.2.2 神经网络拟合GUI

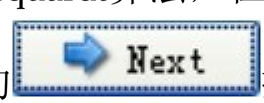
在函数逼近和数据拟合等方面，神经网络得到了广泛的应用。在Simulink中有专门完成数据拟合功能的GUI。

在MATLAB的Command Window窗口输入以下命令。

nftool

运行后可以得到图12-53所示的数据拟合GUI对话框界面。

神经网络数据拟合GUI可以用来收集、建立和训练神经网络，并且利用回归分析和均方误差来评价网络的效果。该GUI采用一个两层前向型神经网络拟合函数，隐含层神经元使用的是Sigmoid函数，输出神经元使用线性神经元。

如果设置足够多的隐含层神经元，神经网络就可以实现多维数据的拟合问题。拟合的训练算法使用了Levenberg-Marquardt算法，在MATLAB的函数名称为trainlm。单击图12-53中的

按钮，出现导入数据的对话框，如图12-54所示。

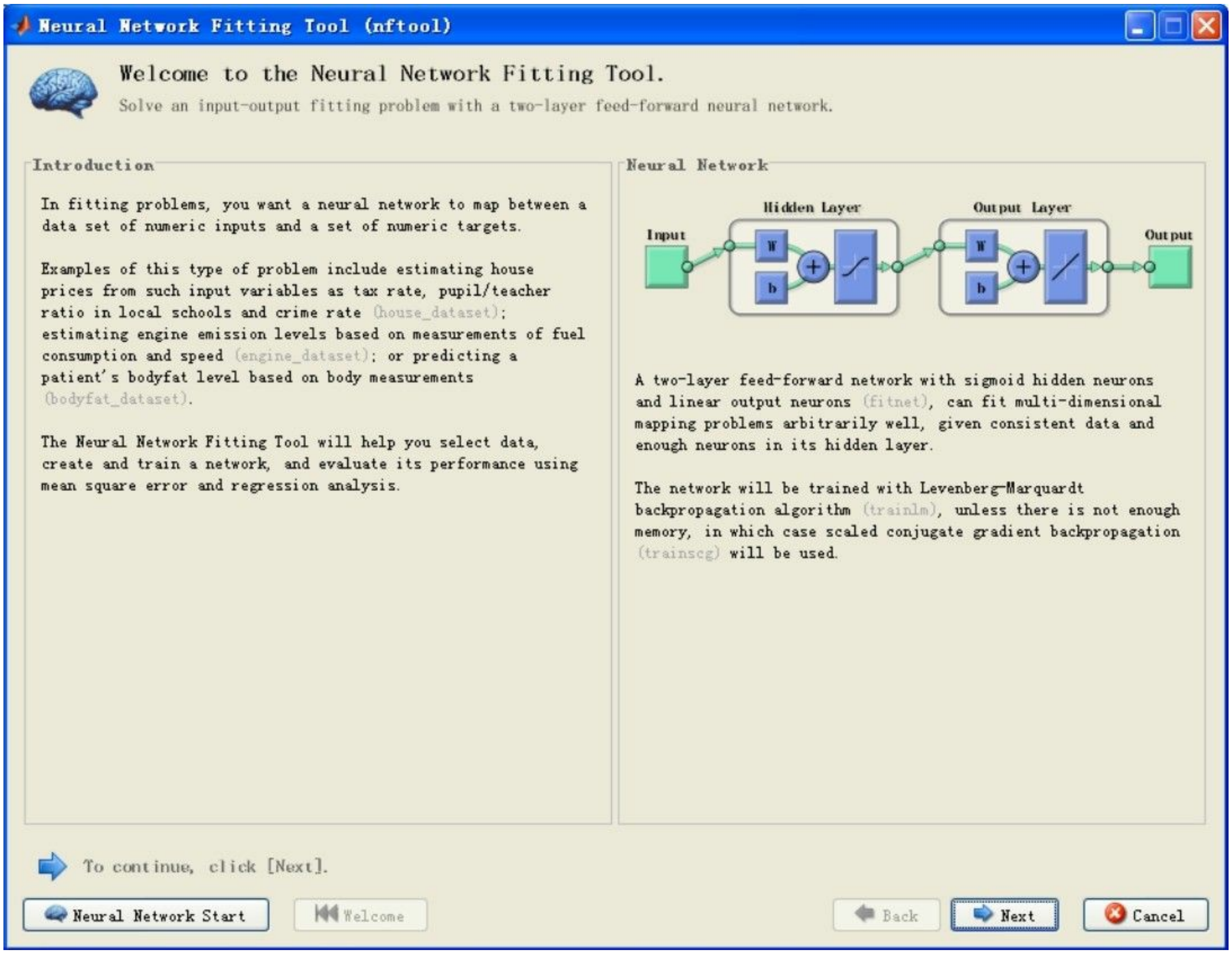


图12-53 数据拟合GUI对话框界面

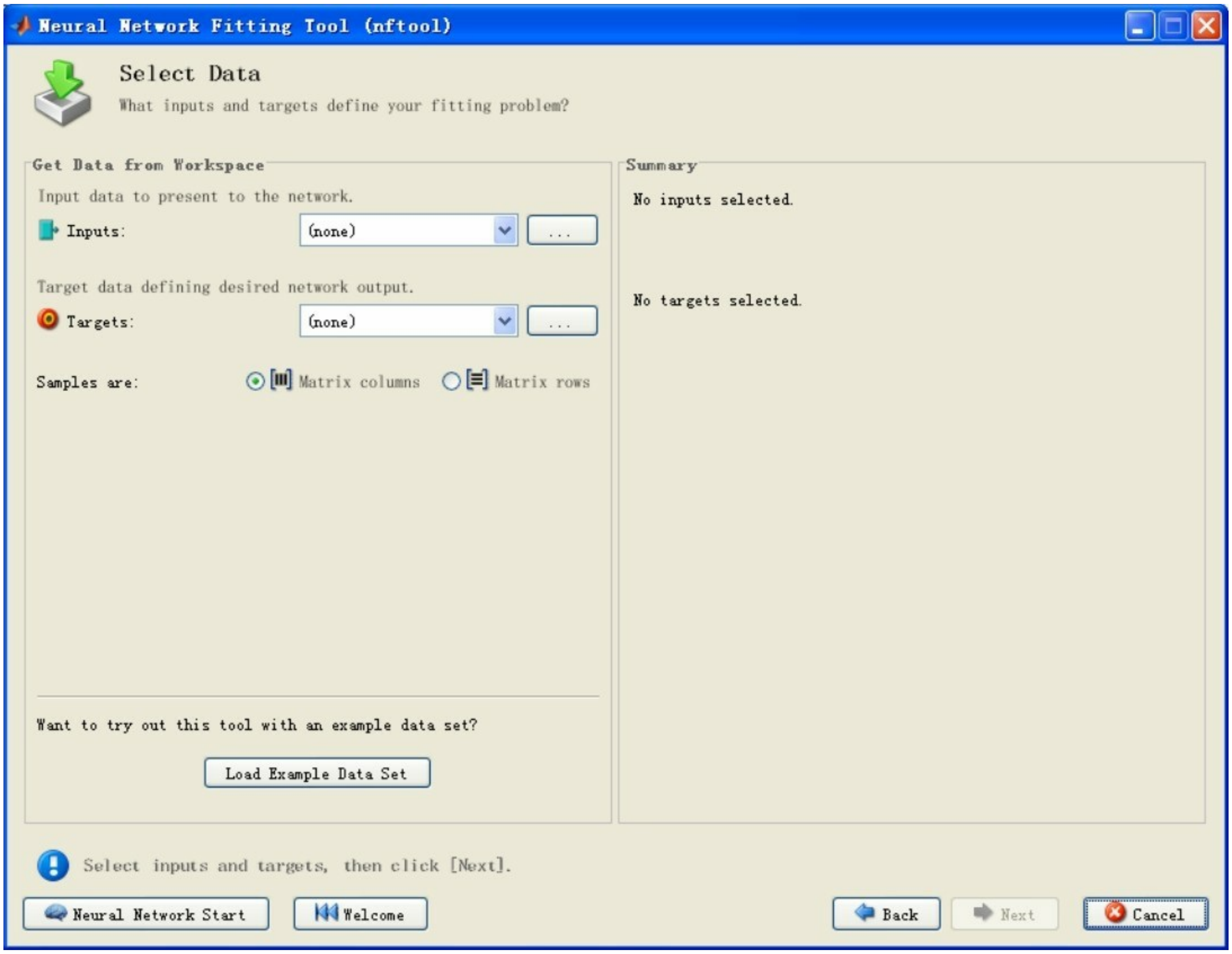


图12-54 导入数据的对话框

从MATLAB的Workspace内可以导入数据，数据分为输入数据和目标数据。在输入数据导入后，数据的大小会自动矩形归一化，即数据大小均在[-1,1]之间。

在MATLAB的Command Window窗口输入以下命令。

x=[1 2 3 4 5];

t=[2 3 5 7 9];

得到x和t两个数据，分别作为神经网络的输入数据和目标数据。

在导入数据的对话框中选择输入数据为x，选择目标数据为t，如图12-55所示。

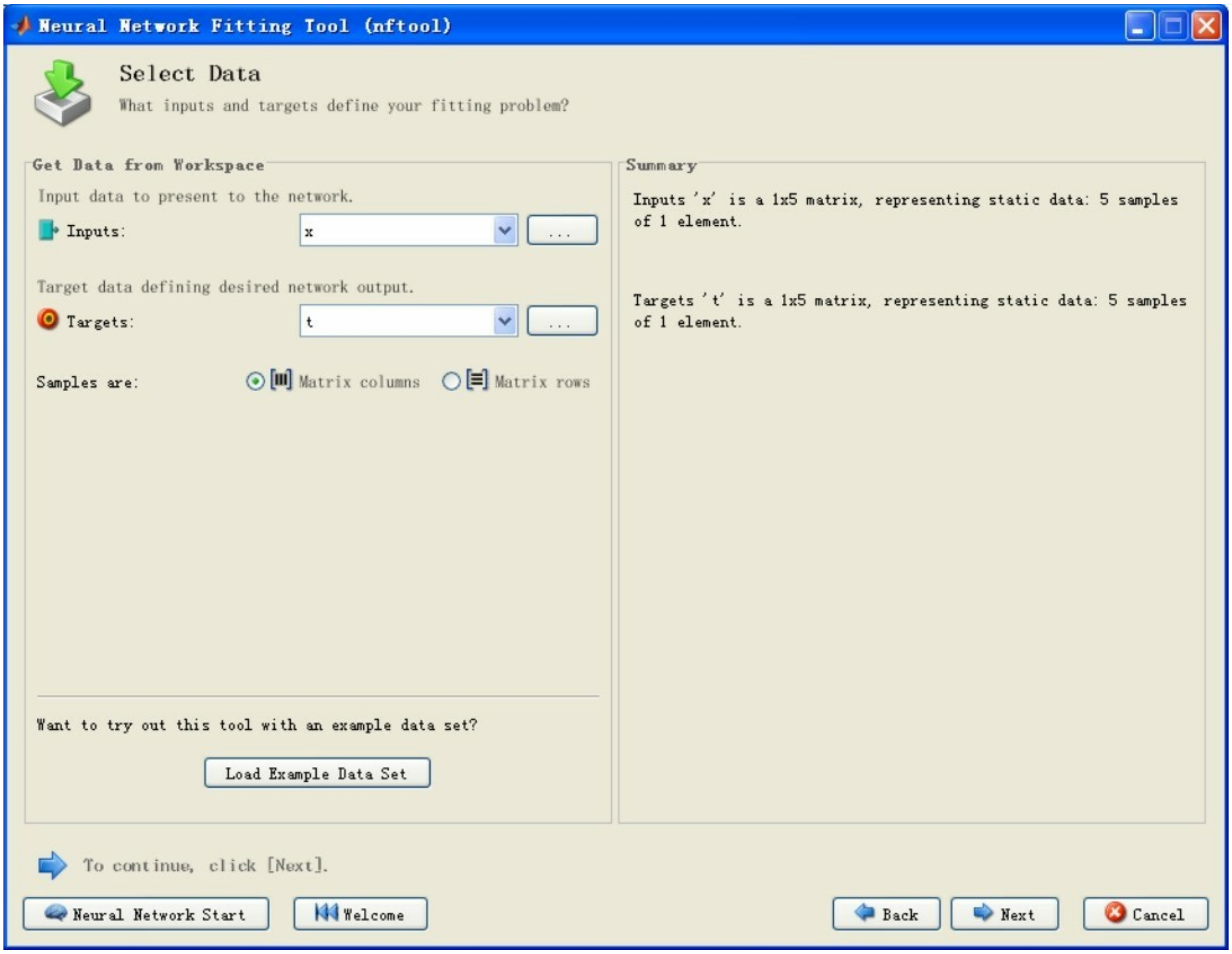
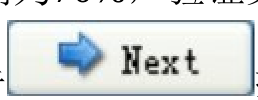


图12-55 选择输入数据和目标数据

单击图 12-55 中的

按钮，可以看到选取验证数据和测试数据的对话框，如图12-56所示。

整个数据分为训练数据、验证数据和测试数据。其中，训练数据是用来训练神经网络的样本数据，目的是为了使神经网络对训练数据的特征进行学习。验证数据同样是来自网络训练，目的是为了在训练过程中提高网络的泛化能力。

图12-56中左边的3个百分比是对应总的数据来说的，即训练数据、验证数据和测试数据分别占整个数据的比例。从图中可以看出，训练数据占的比例为70%，验证数据占的比例为15%，测试数据占的比例为15%。单击

按钮，得到图12-57所示的选择网络结构对话

框。

该对话框能设置神经网络的隐含层神经元个数。由于使用了三层的前向型神经网络，所以输入和输出数据确定后，可以调整的只有隐含层的神经元数目。在1～100中，隐含层神经元数目设置为10。

提示：如果在测试过程中，发现拟合的效果不好，可以回到这个对话框来重新调整隐含层神经元数目。

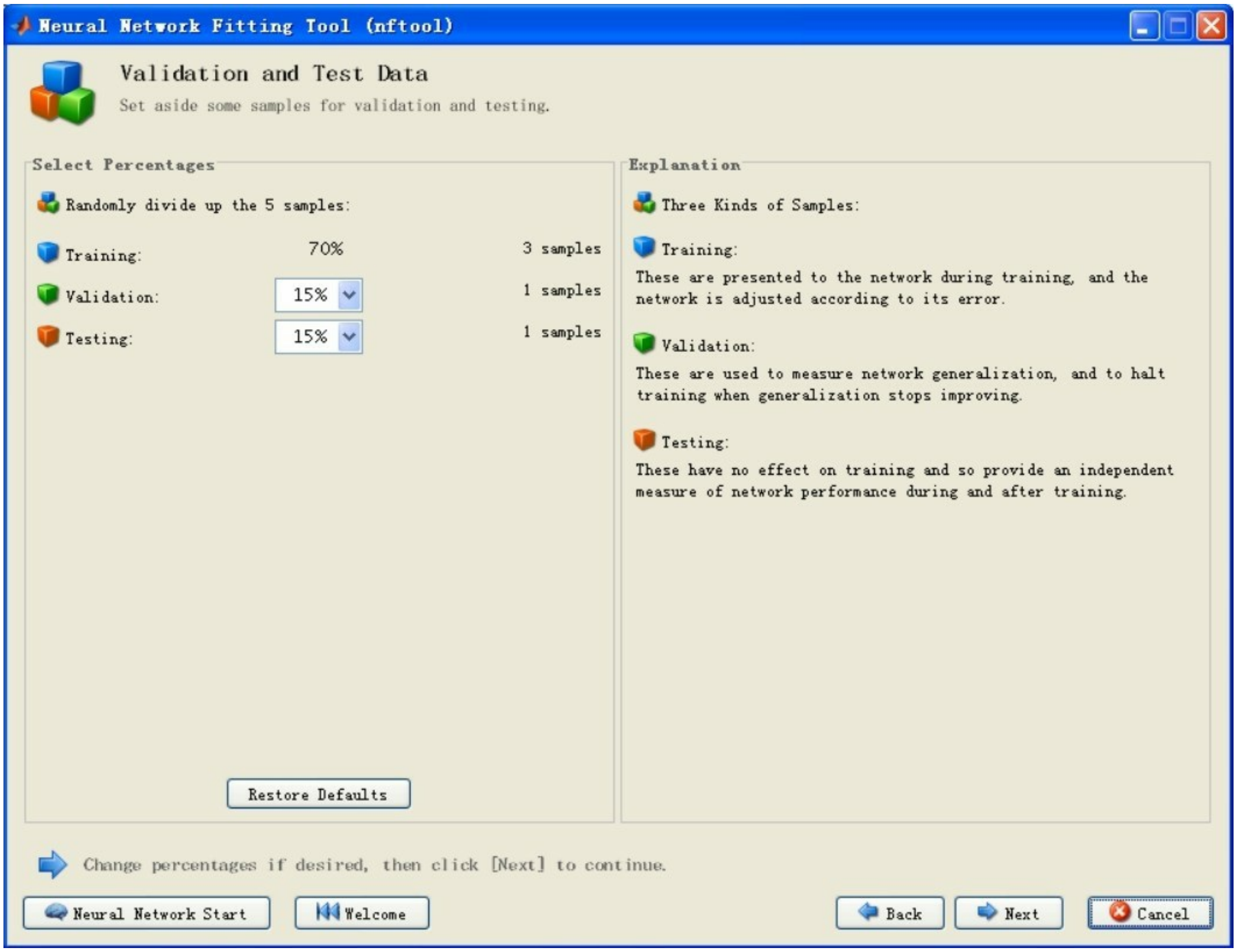


图12-56 选取验证数据和测试数据对话框

单击图12-57中的

按钮，得到网络训练对话框，如图12-58所示。

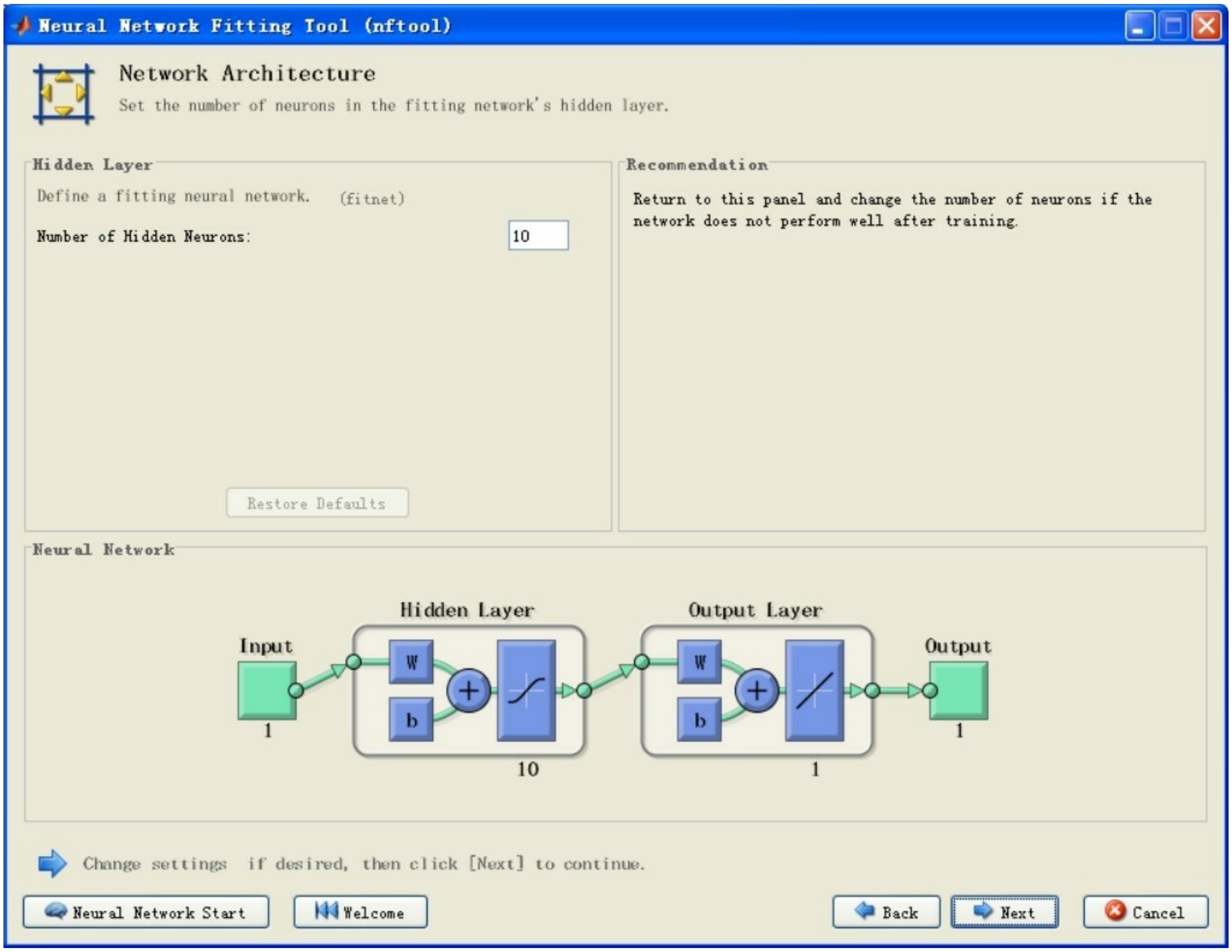
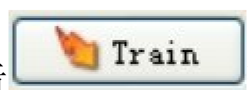


图12-57 选择网络结构对话框

在图12-58中，单击

按钮，可以对设计的神经网络进行训练。神经网络的训练窗口如图12-59所示。训练后得到的网络训练对话框如图12-60所示。

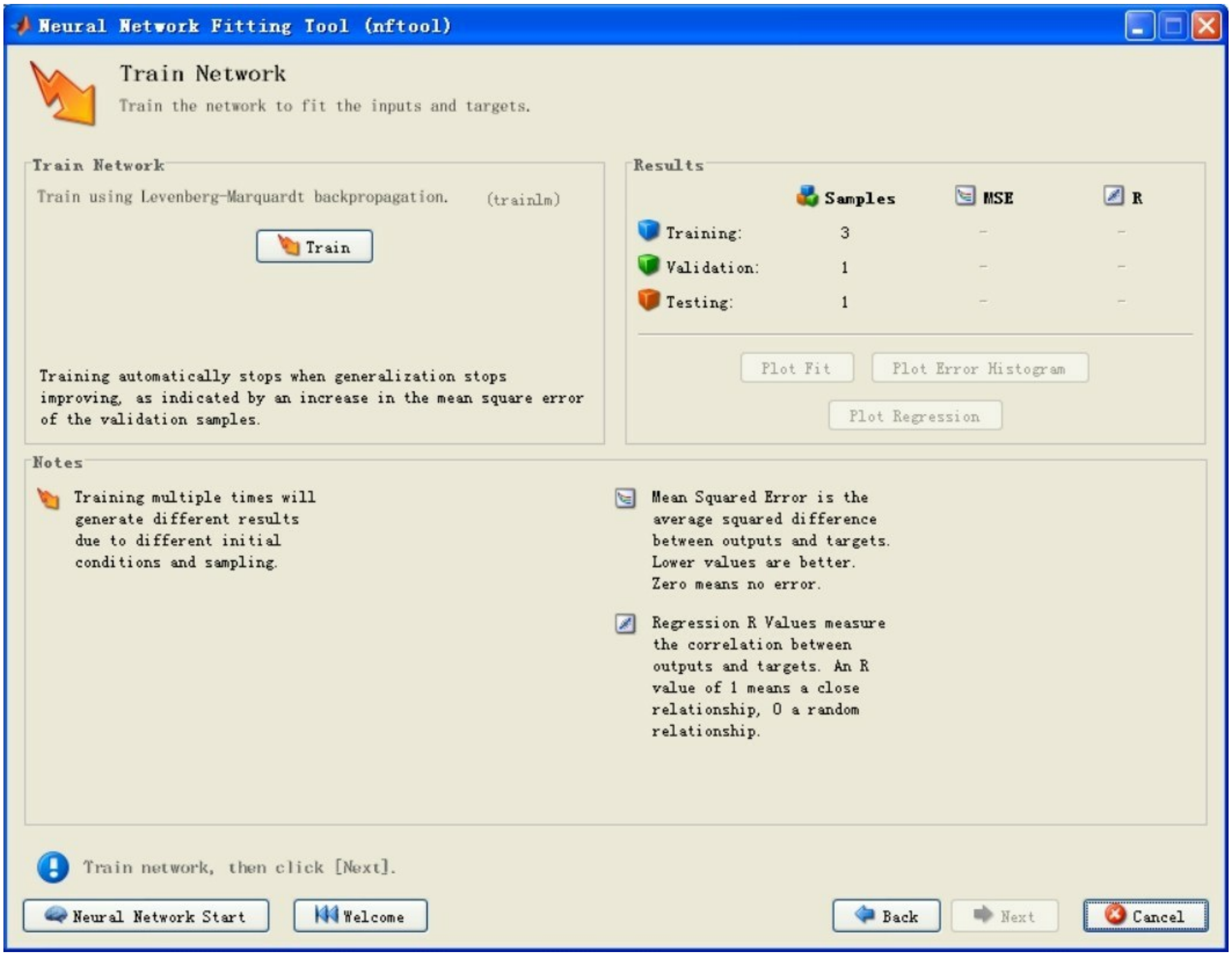


图12-58 网络训练对话框

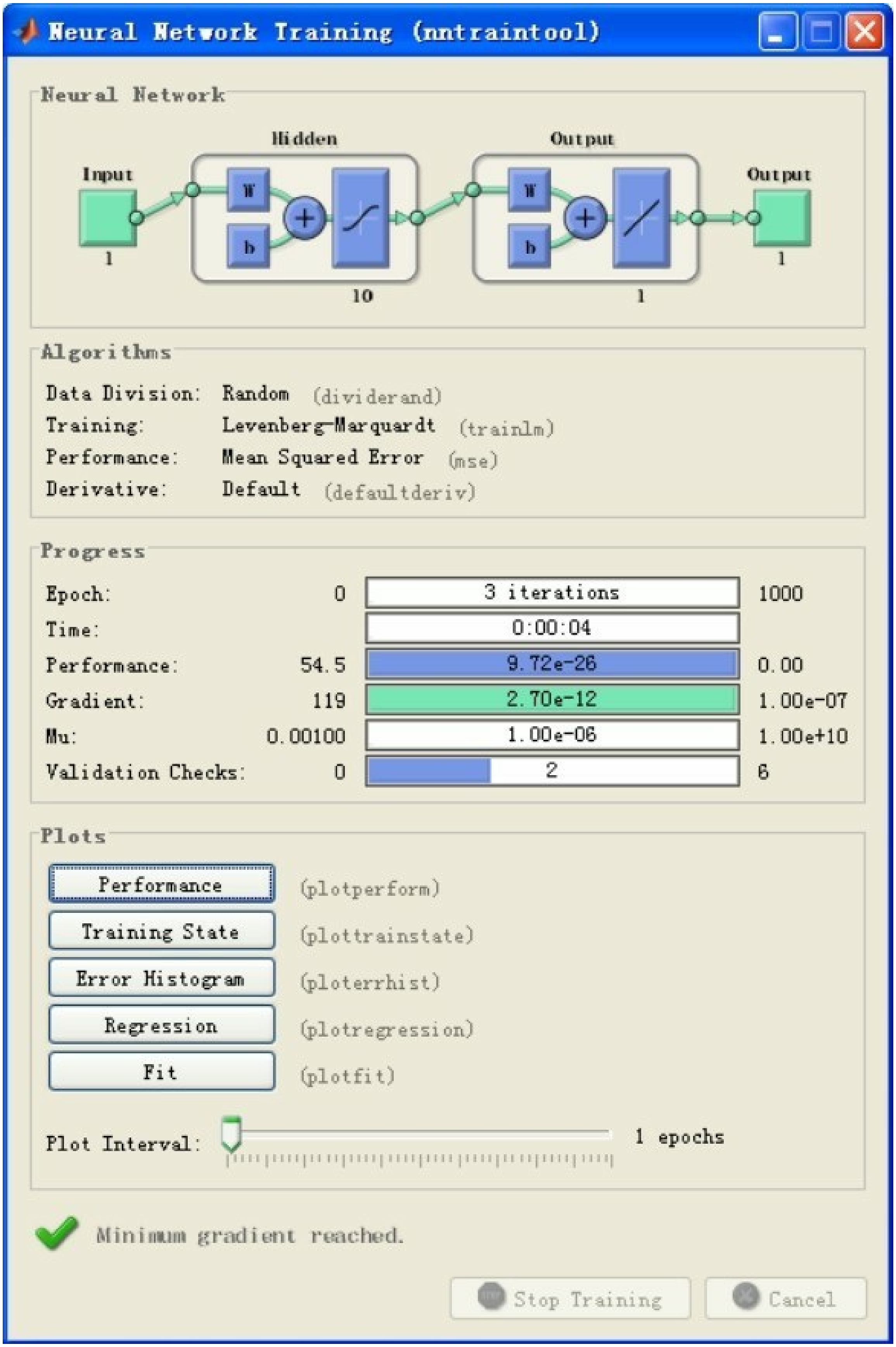

Neural Network Training (nntraintool)
Neural Network
Hidden
Output
Input
Output
W
b
1
10
1
1
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)
Progress
Epoch: 0 3 iterations 1000
Time: 0:00:04
Performance: 54.5 9.72e-26 0.00
Gradient: 119 2.70e-12 1.00e-07
Mu: 0.00100 1.00e-06 1.00e+10
Validation Checks: 0 2 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)
Plot Interval: 1 epochs
Minimum gradient reached.
Stop Training
Cancel

图12-59 神经网络训练窗口

从图 12-60 可以看出，在网络训练结束后，神经网络训练对话框中的 Plot Fit 、 Plot Error Histogram 、 Plot Regression 按钮由不可用状态变为可用状态，这 3 个按钮的作用是画出训练过程中的相关参数曲线。如单击 Plot Fit 按钮，可以得到图12-61所示的神经网络训练输入数据与目标数据比较图及误差曲线。

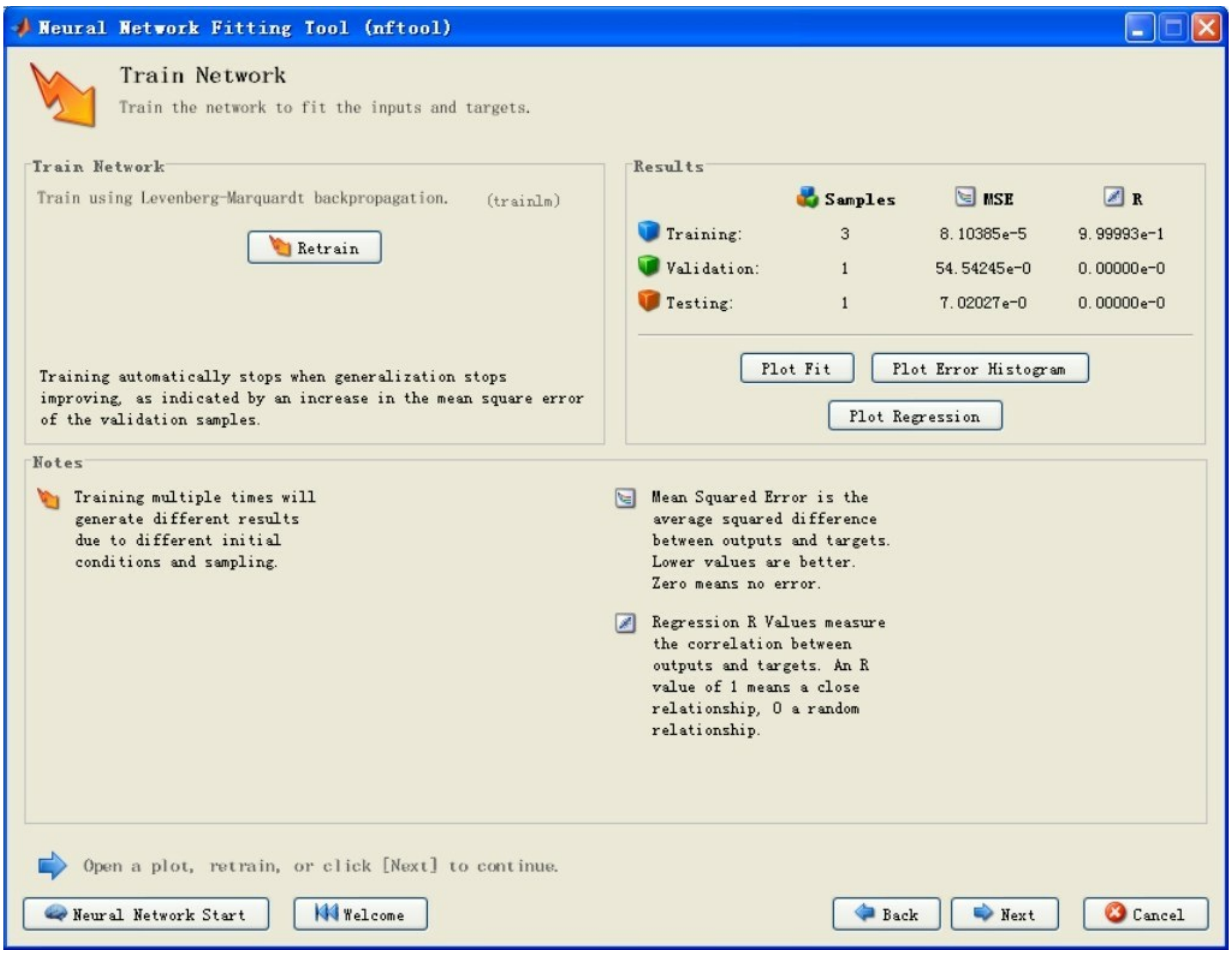


图12-60 神经网络训练后的对话框

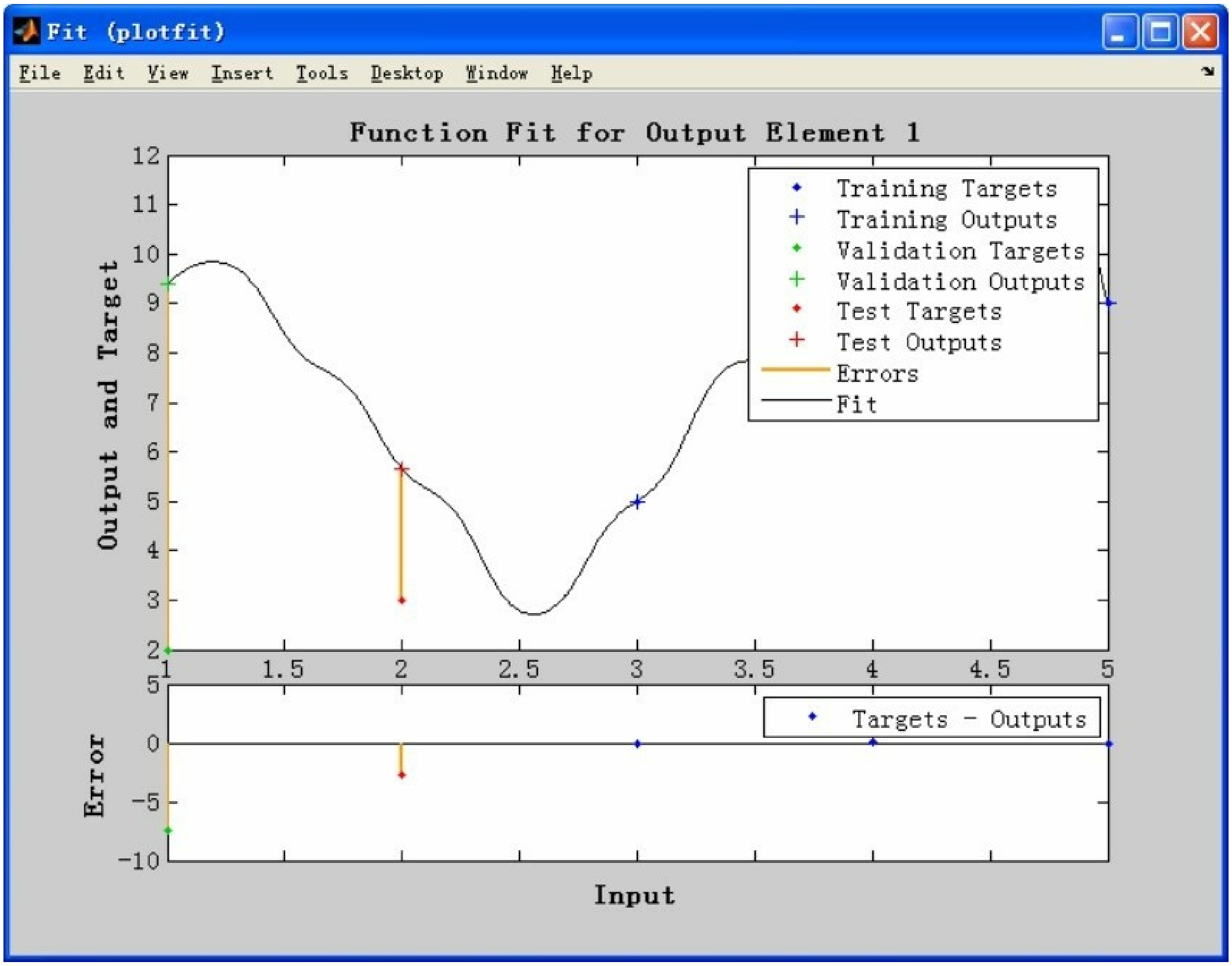
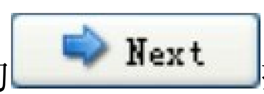
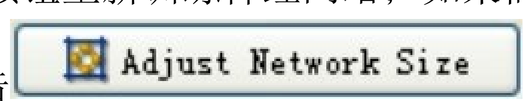


图12-61 神经网络训练输入数据与目标数据比较图及误差曲线

注意：在网络训练过程中，如果网络的泛化效果停止提高时，神经网络的训练会自动停止。选取的初始条件和样本数据不同，导致训练的结果也会不同。

训练完成后，单击图12-60中的

按钮，得到修正神经网络训练的对话框，如图12-62所示。当神经网络拟合效果不好时，可以单击图12-62中

按钮重新训练神经网络；如果需要调整神经网络神经元数目，可以单击

按钮；如果需要调整输入或目标数据，可以单击



按钮。

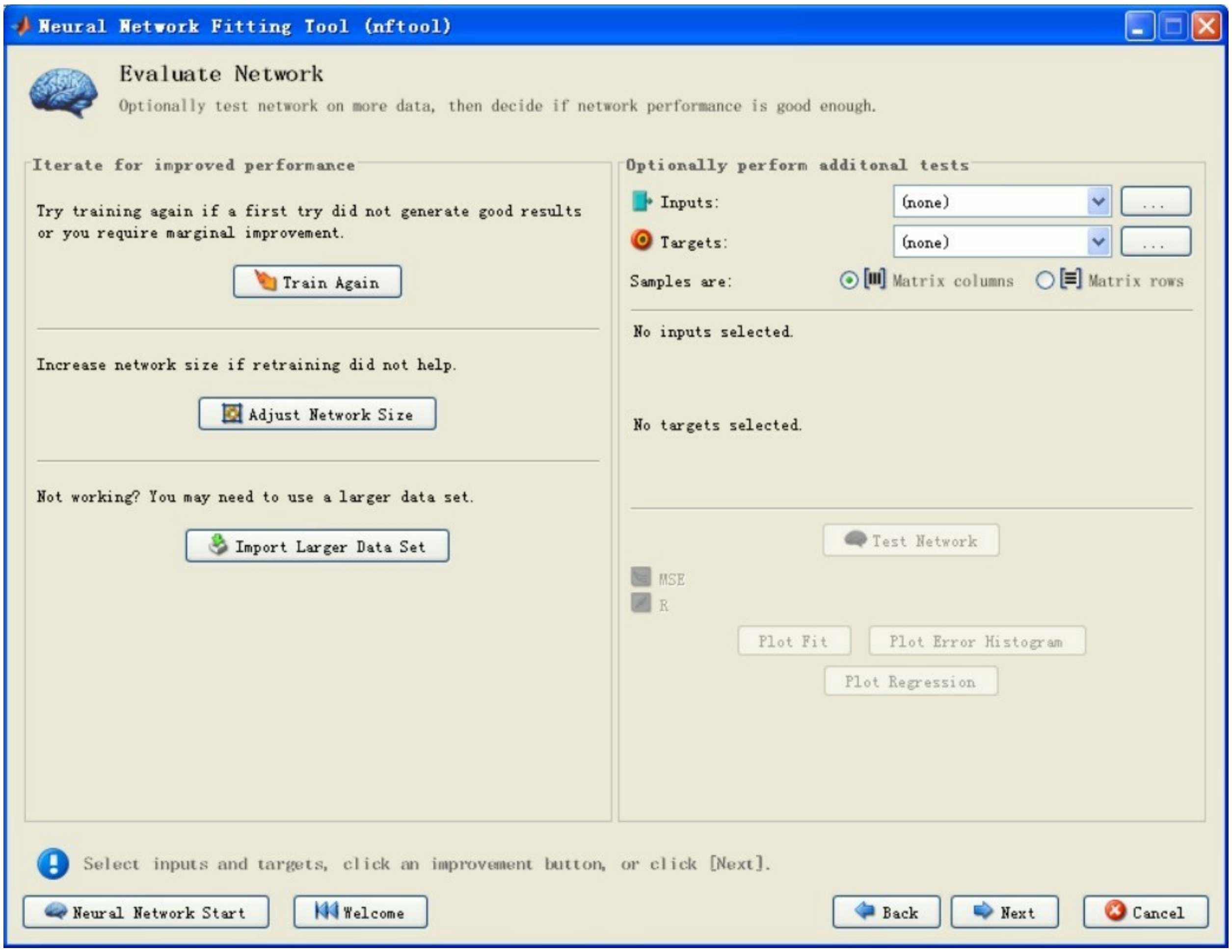
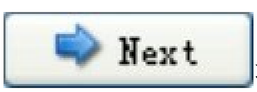


图12-62 修正神经网络训练的对话框

如果认为网络的拟合效果达到预期目标，单击图12-62中的



按钮，得到保存数据和神经网络的对话框，如图12-63所示。

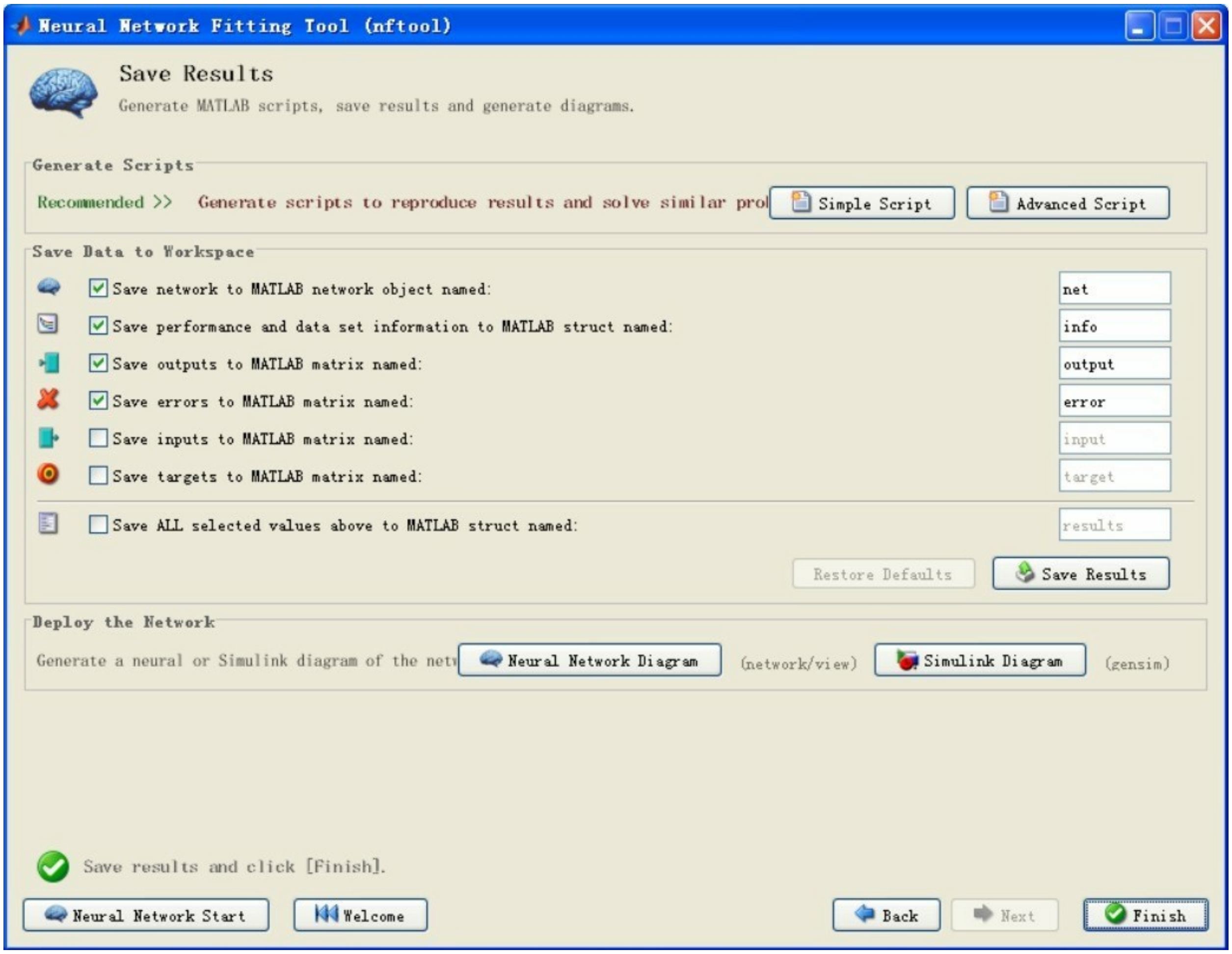


图12-63 保存数据和神经网络的对话框

这个对话框运行存储输入、输出、误差和网络结构等与训练相关的数据。

在图12-63中单击

按钮，MATLAB会出现图12-64所示的神经网络确认保存警告，单击

按钮，就将训练好的神经网络保存成功。

将训练好的网络保存以后，如果有新的需要拟合的数据，可以通过调用保存好的神经网络进行拟合，从而节省建立和训练网络的时间。

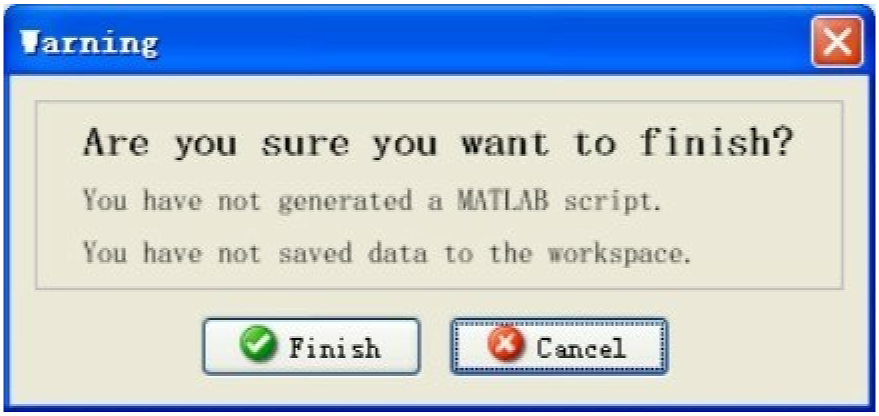


图12-64 确认保存警告

12.2.3 神经网络模式识别GUI

在模式识别方面，神经网络也得到了广泛的应用。在Simulink中有专门完成数据拟合功能的GUI。

在MATLAB的Command Window窗口输入以下命令。

nprtool

运行后可以得到图12-65所示的数据拟合GUI对话框界面。

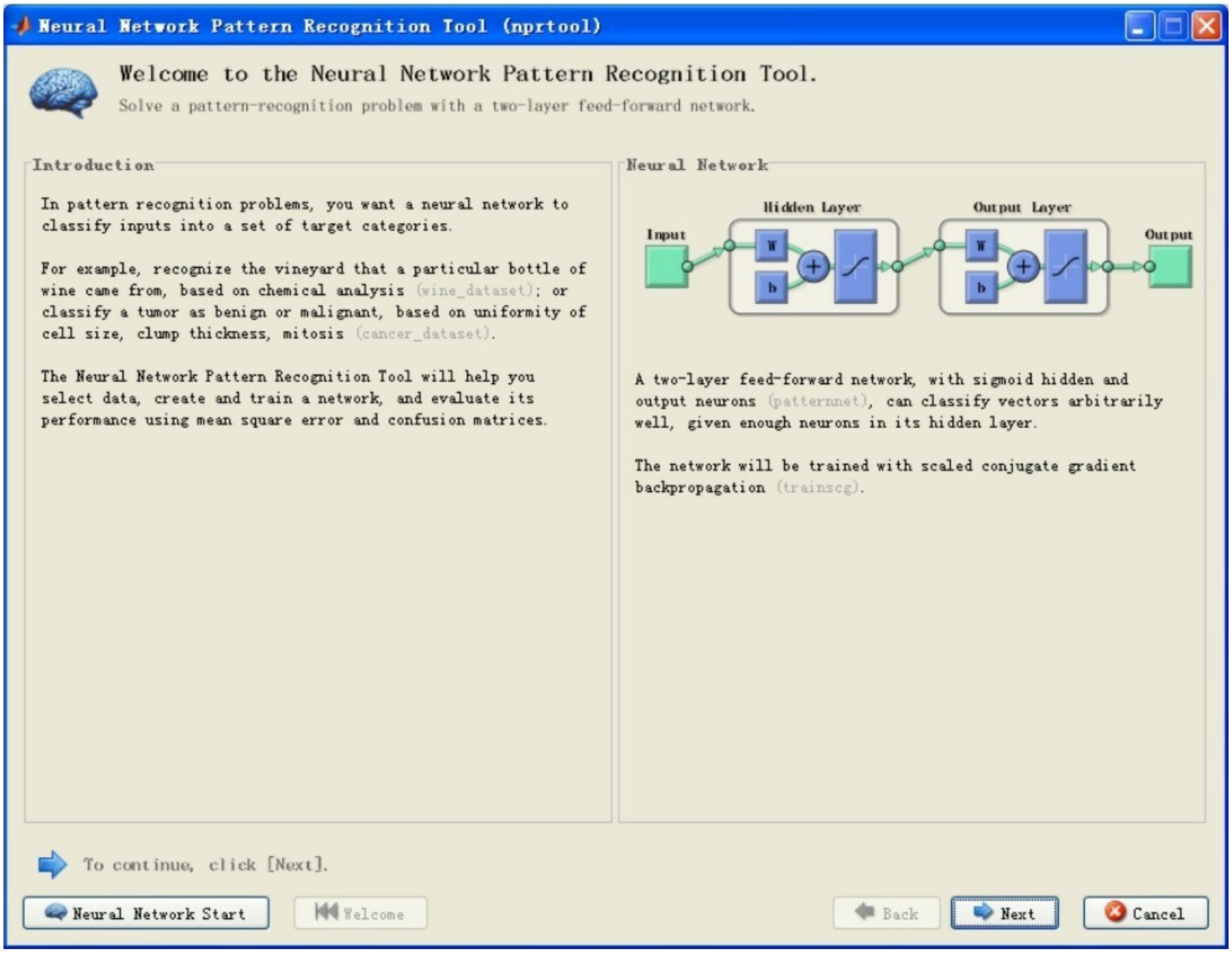


图12-65 数据拟合GUI对话框界面

从图 12-65 中可以看出，神经网络模式识别用来收集、建立和训练神经网络，并且利用均方误差等来评价网络的效果。

一般用来进行模式识别的是一个两层的前向型神经网络，隐含层神经元和输出层神经元使用的都是Sigmoid函数。

神经网络解决模式识别问题都是通过建立一种神经网络来对已有的分类目标数据进行学习，训练好网络后，再将其用于数据的分类。

模式识别神经网络的训练使用了量化连接梯度训练函数，即trainscg算法。单击图12-65中的

按钮，出现神经网络导入数据的对话框。

在MATLAB的Command Window窗口输入以下命令。

x=[1 2 3 4 5];

t=[2 3 5 7 9];

得到x和t两个数据，分别作为神经网络的输入数据和目标数据。

在导入数据的对话框中选择输入数据为x，选择目标数据为t，如图12-66所示。

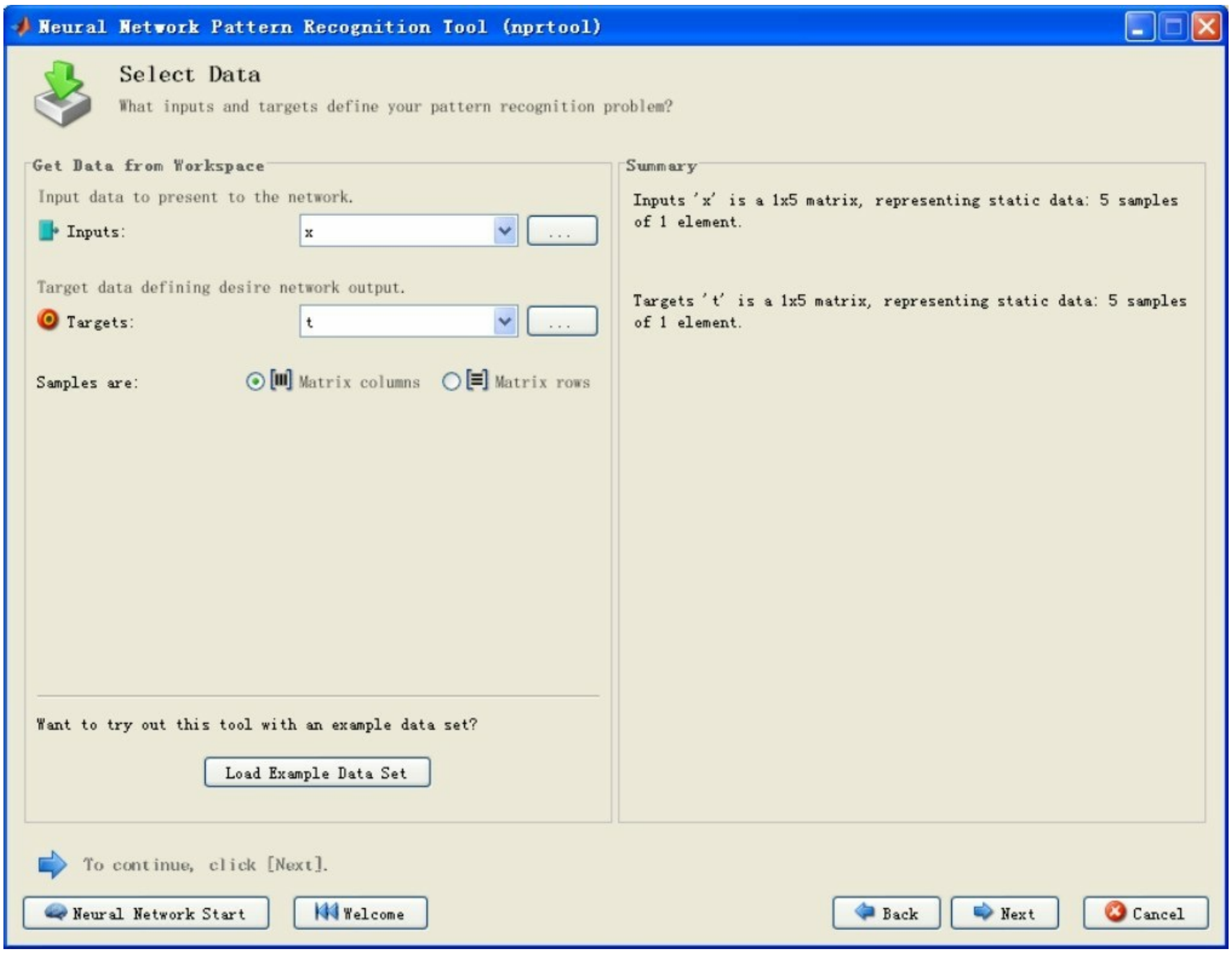
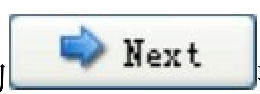


图12-66 选择神经网络的输入数据和目标数据的对话框

单击图 12-66 中的

按钮，可以看到选取验证数据和测试数据的对话框，如图12-67所示。

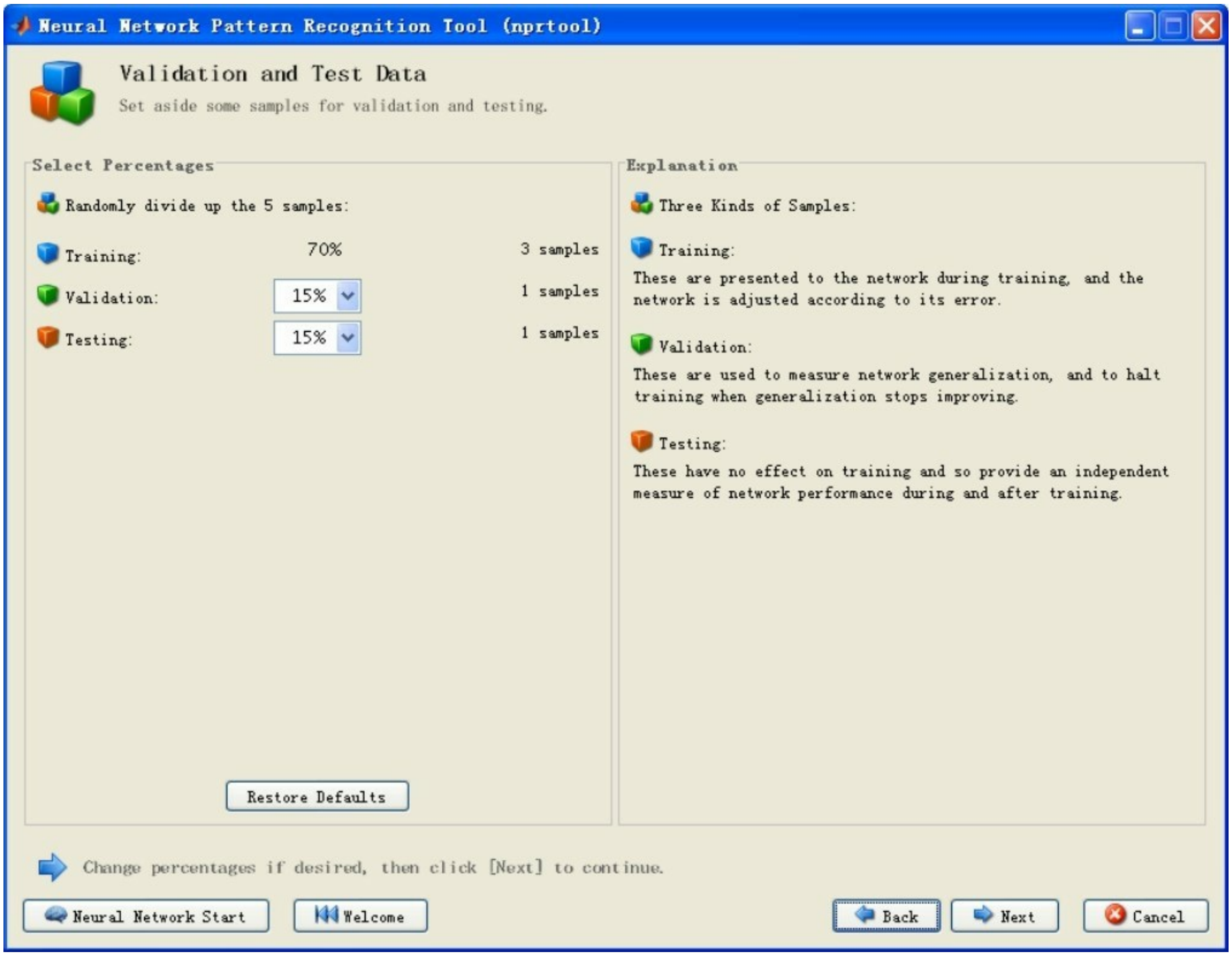


图12-67 选取验证数据和测试数据对话框

在图12-67中单击

按钮，隐含层神经元个数选为20个，得到图12-68所示的选择网络结构对话框。

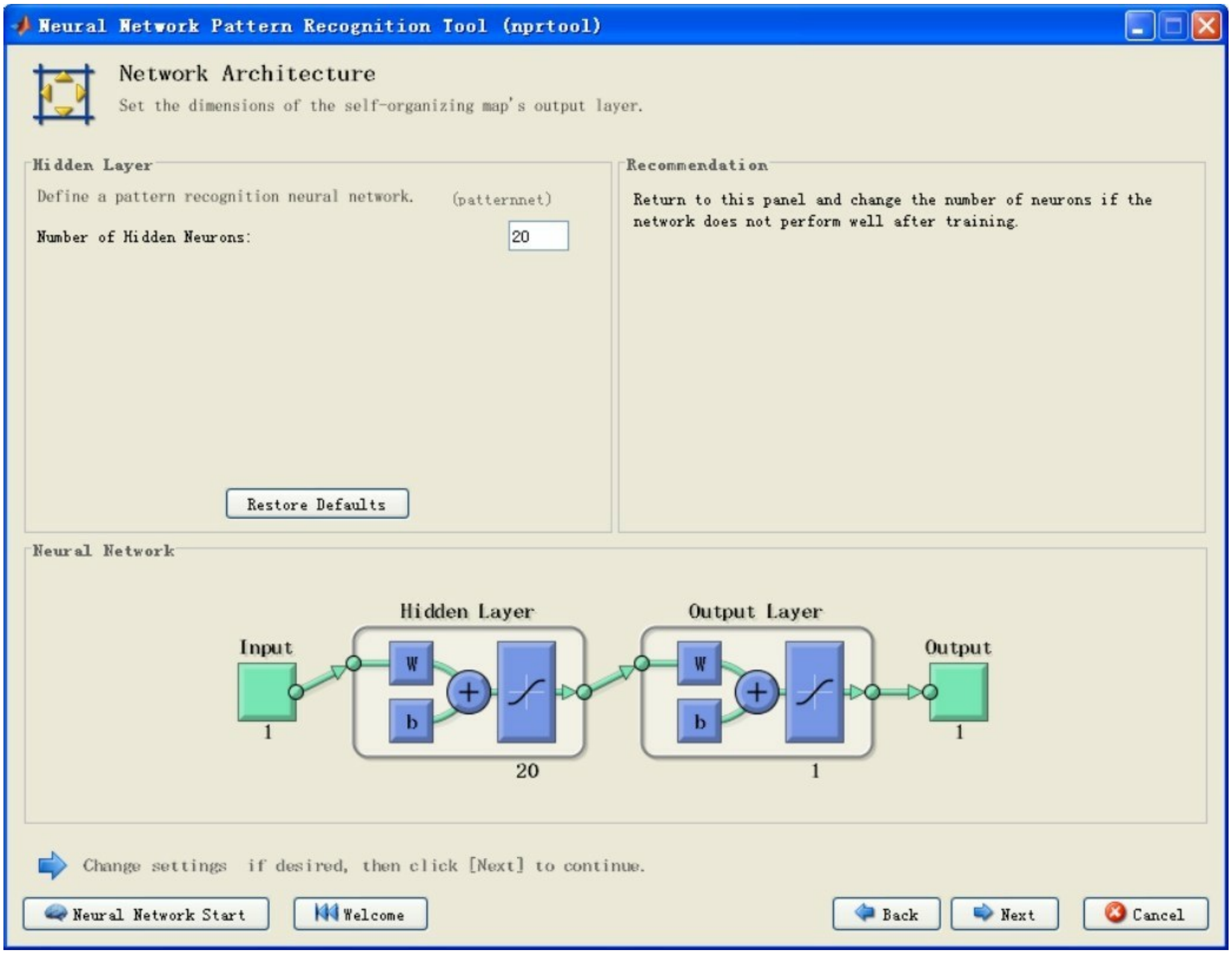
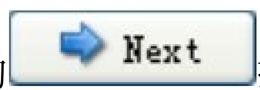


图12-68 选择网络结构对话框

单击图12-68中的按钮，得到网络训练对话框，如图12-69所示。

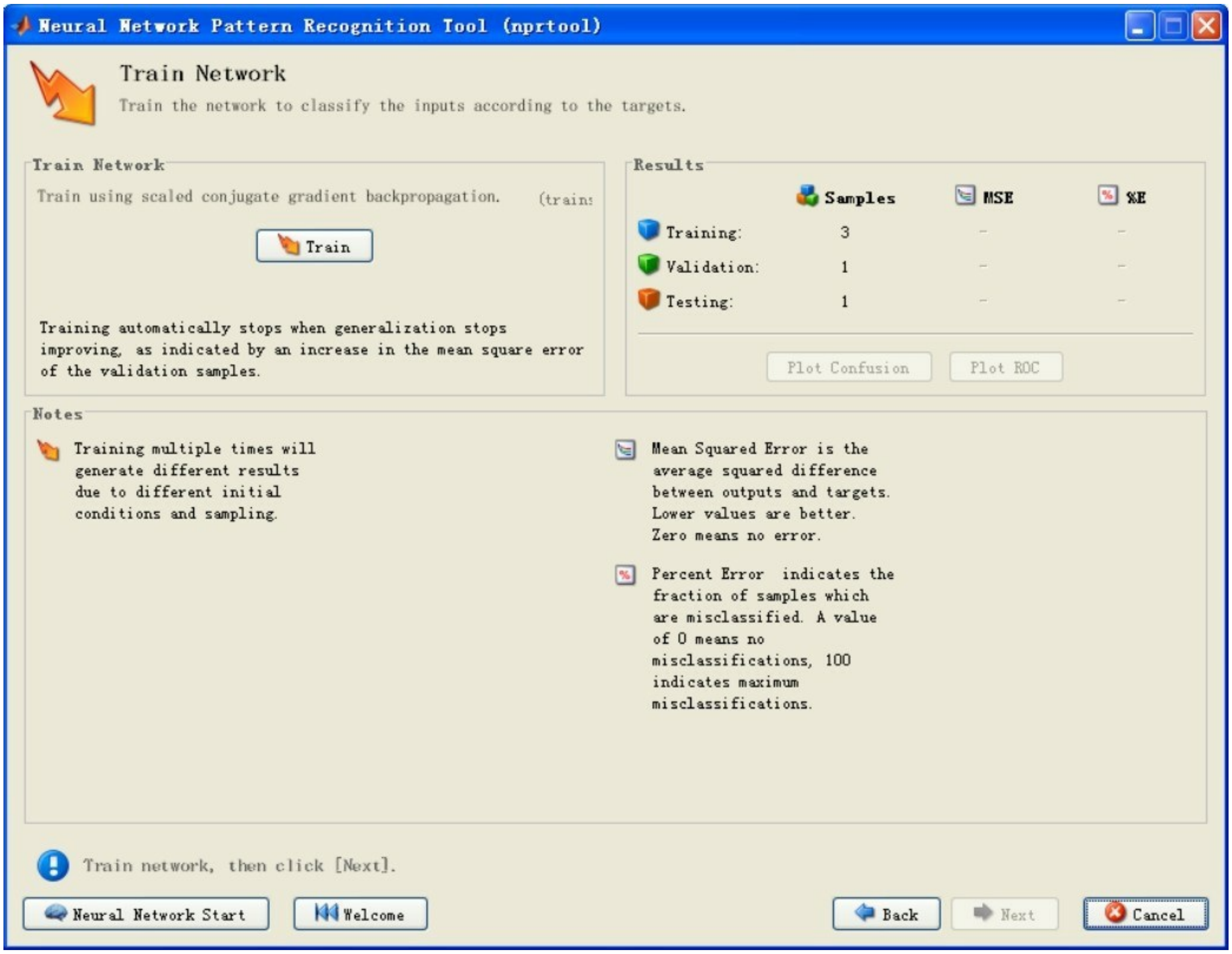
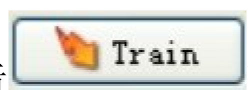


图12-69 网络训练对话框

在图12-69中，单击按钮，可以对设计的神经网络进行训练。训练后得到的网络训练对话框如图12-70所示。

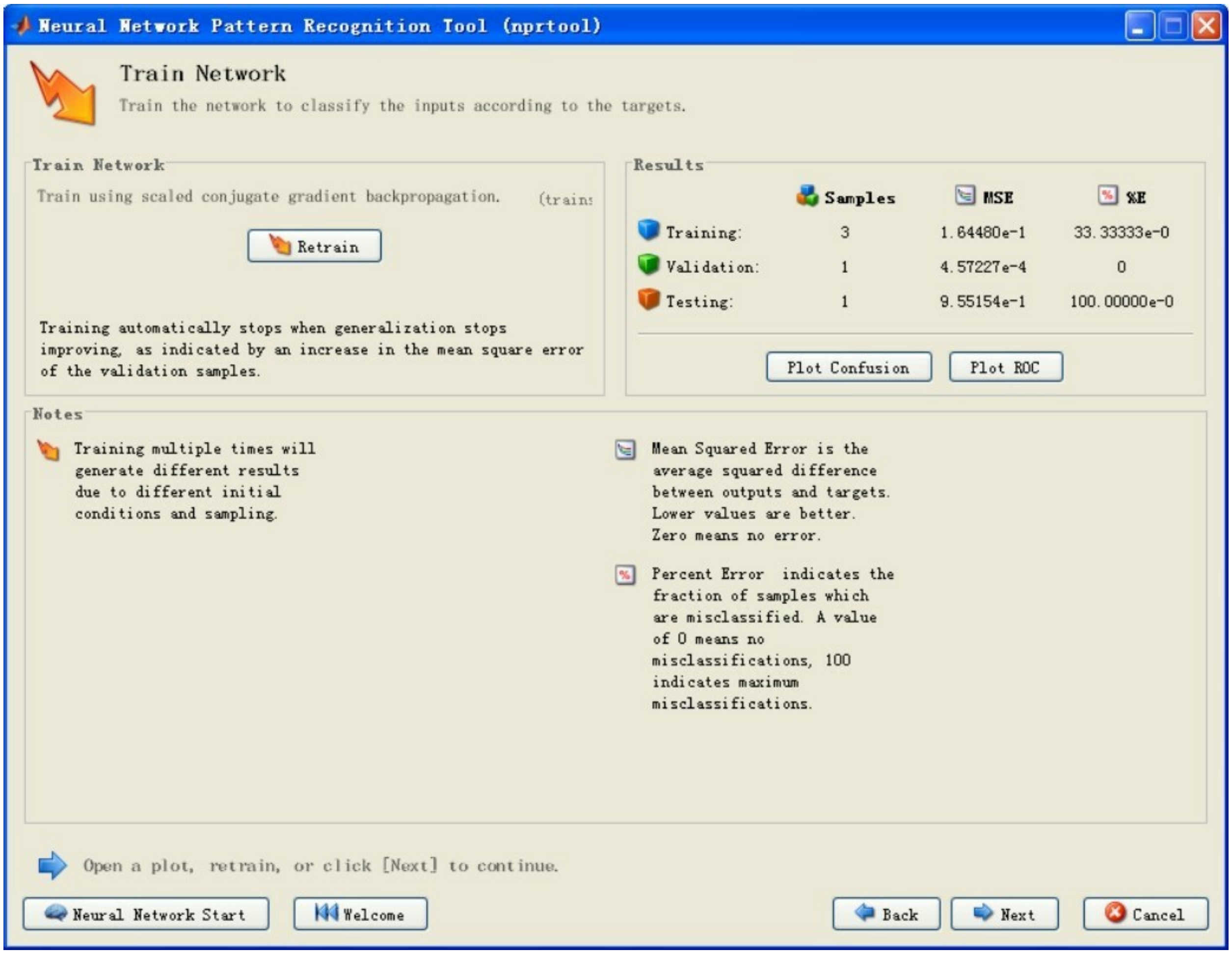


图12-70 神经网络训练后的对话框

从图 12-70 可以看出，在网络训练结束后，神经网络训练对话框中的

和

按钮由不可用状态变为可用状态，这两个按钮的作用是查看神经网络的分类结果。

如单击

按钮，可以得到图12-71所示的神经网络对各种数据（训练数据、测试数据、验证数据和所有数据）的分类结果。

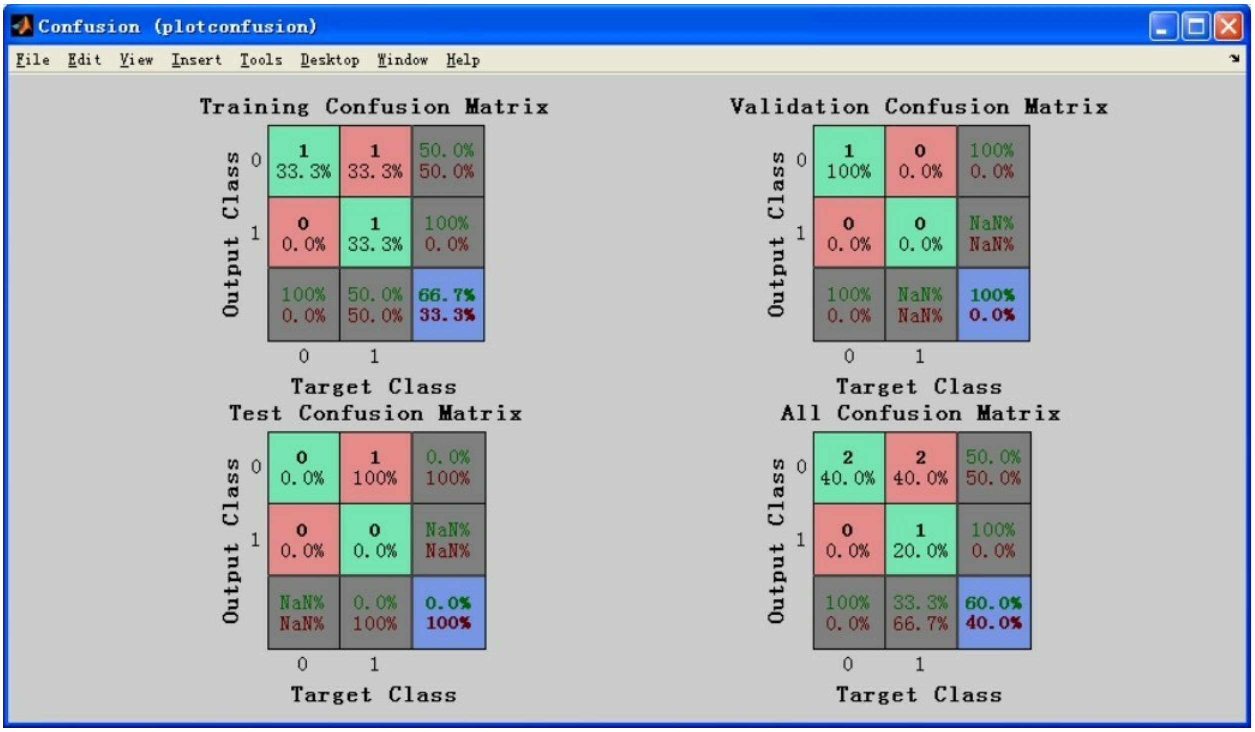


图12-71 神经网络训练输入数据与目标数据比较图及误差曲线

Confusion Matrix（匹配矩阵）是一种展示分类效果好坏的矩阵。匹配矩阵把所有正确和错误的分类信息都归到一个表里。ROC曲线是反映敏感性和特异性连续变量的综合指标。

如果单击

按钮，可以得到图12-72所示的ROC曲线。

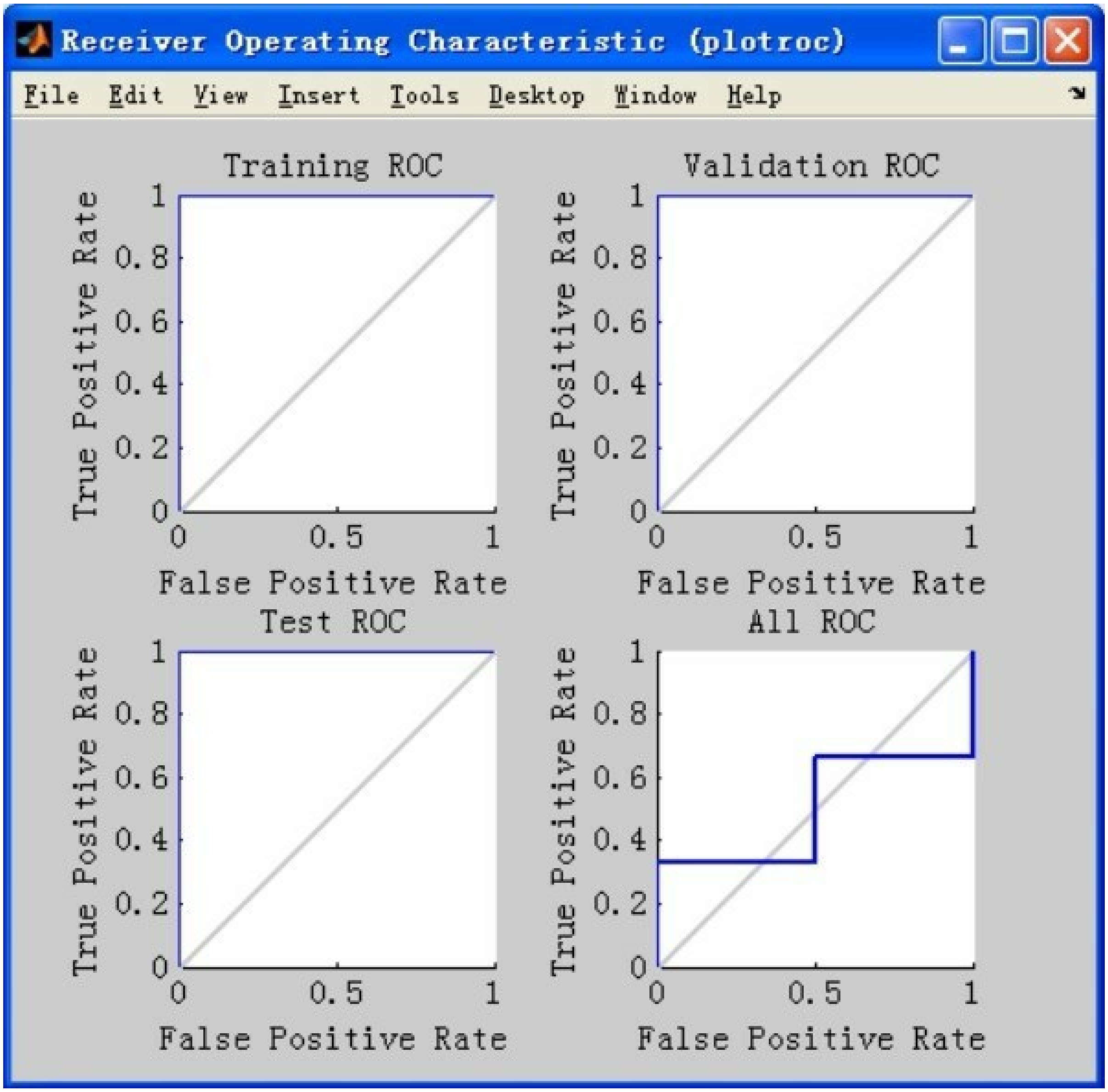
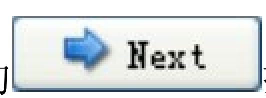
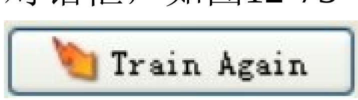


图12-72 ROC曲线

ROC曲线真阳性率为纵坐标，假阳性率为横坐标。在坐标上由无数个临界值求出的无数对真阳性率和假阳性率作图构成，计算ROC曲线下面的AUC来评价分类效率。

训练完成后，单击图12-70中的

按钮，得到修正神经网络训练的对话框，如图12-73　所示。当神经网络拟合效果不好时，可以单击图中

按钮重新训练神经网络；如果需要调整

神经网络神经元数目，可以单击

按钮；如果需要调整输入或目标数据，可以单击


按钮。

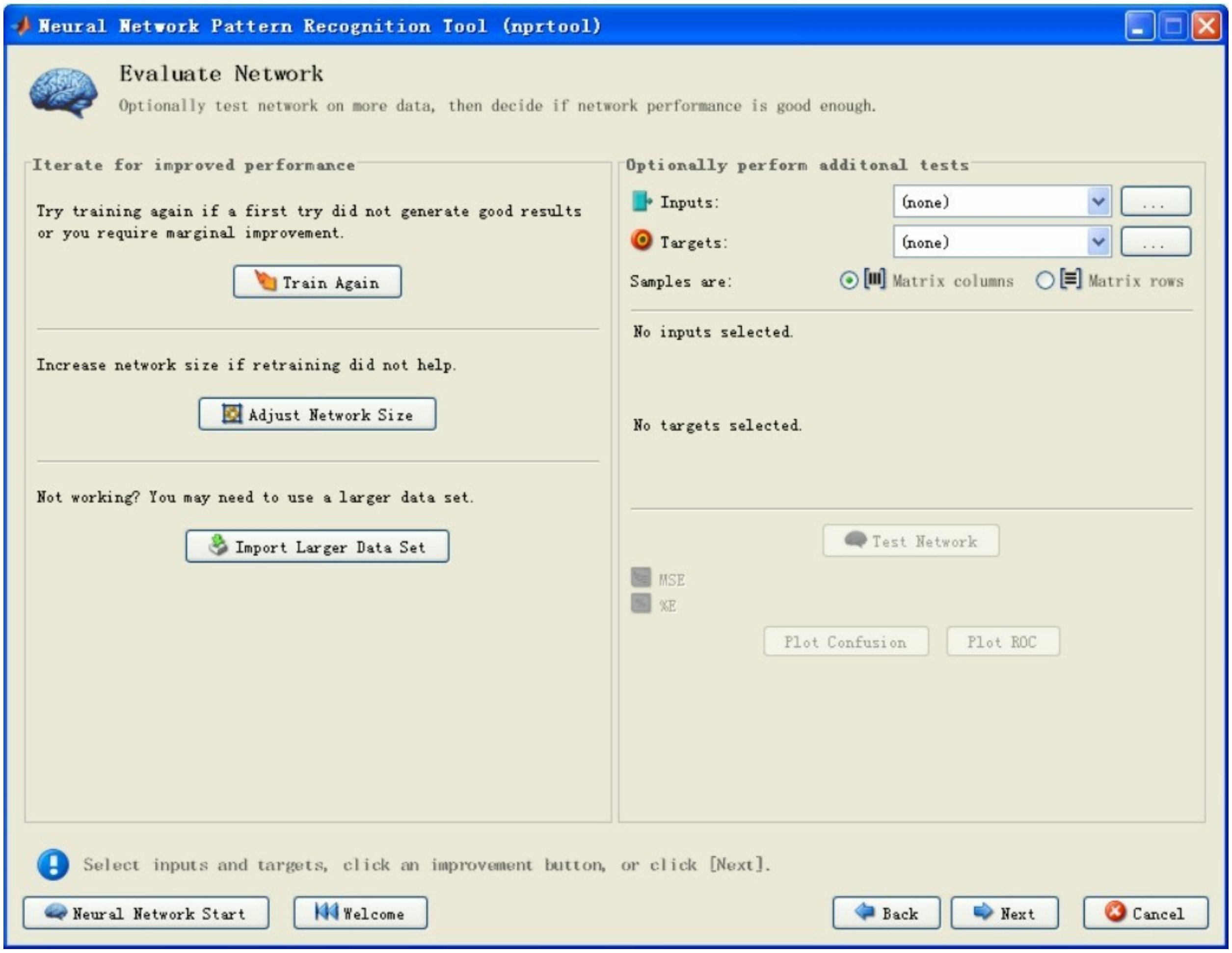


图12-73 修正神经网络训练的对话框

如果需要测试训练好的网络是否达到预期目标，可以设置输入和目标数据，之后图12-73中的

按钮被激活，如图12-74所示。

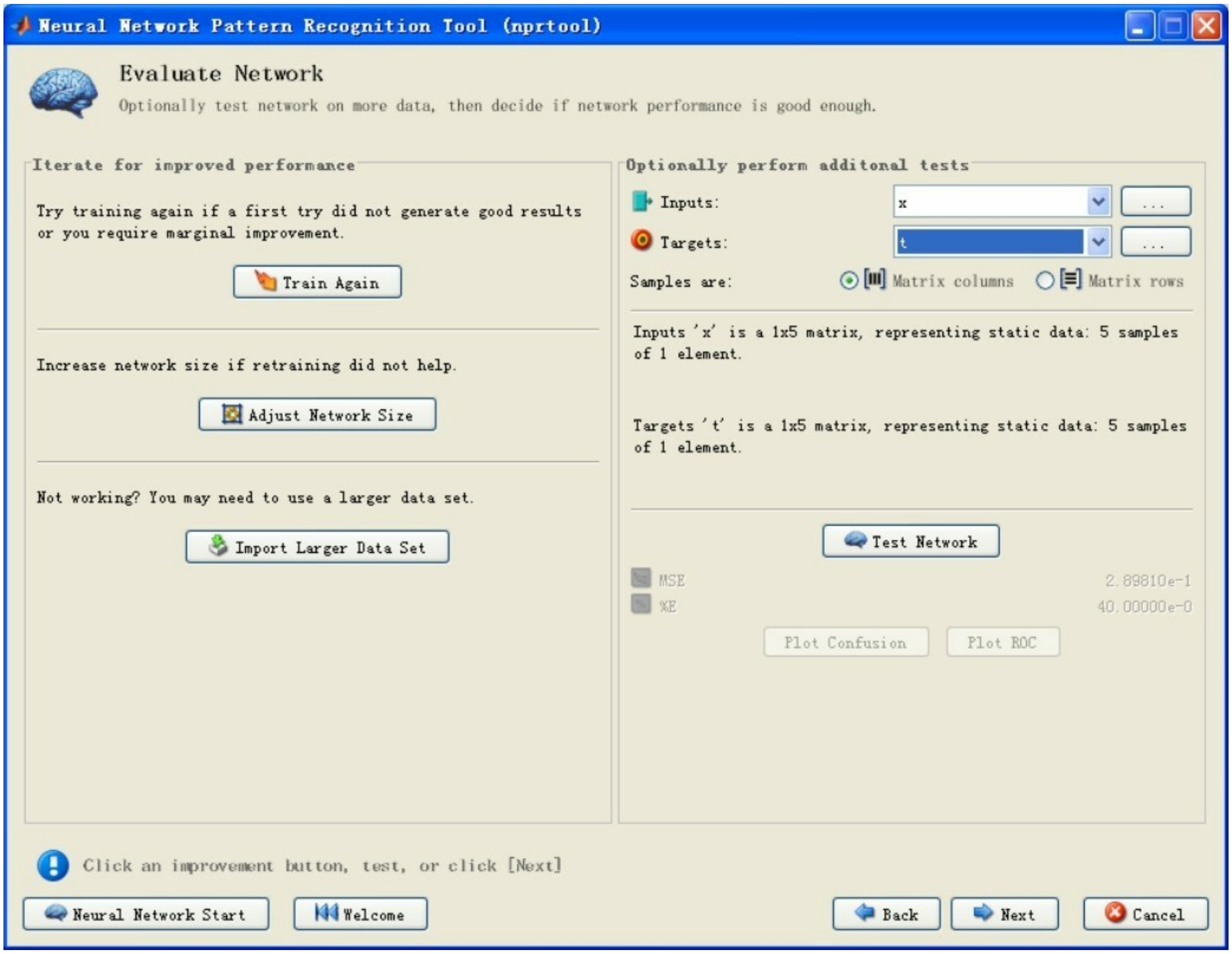


图12-74 Test Network按钮被激活界面

单击

按钮，得到训练好的神经网络测试结果如图12-75所示。

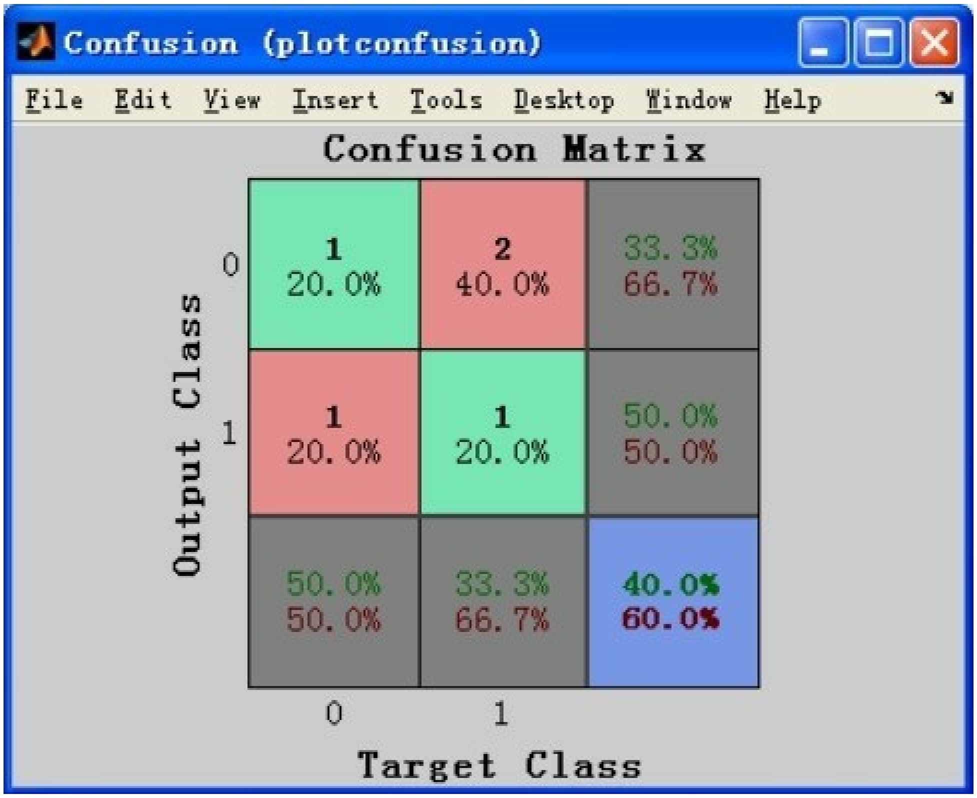
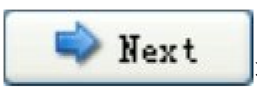


图12-75 训练好的神经网络测试结果

如果认为网络的拟合效果达到预期目标，单击图12-74中的

按钮，得到保存数据和神经网络的对话框，如图12-76所示。

这个对话框运行存储输入、输出、误差和网络结构等与训练相关的数据。

在图 12-76 中单击

按钮， MATLAB 会出现神经网络确认保存警告，单击

按钮，就将训练好的神经网络保存成功。

12.2.4 神经网络聚类GUI

在模式识别方面，神经网络也得到了广泛的应用。在Simulink中有

专门完成数据拟合功能的GUI。

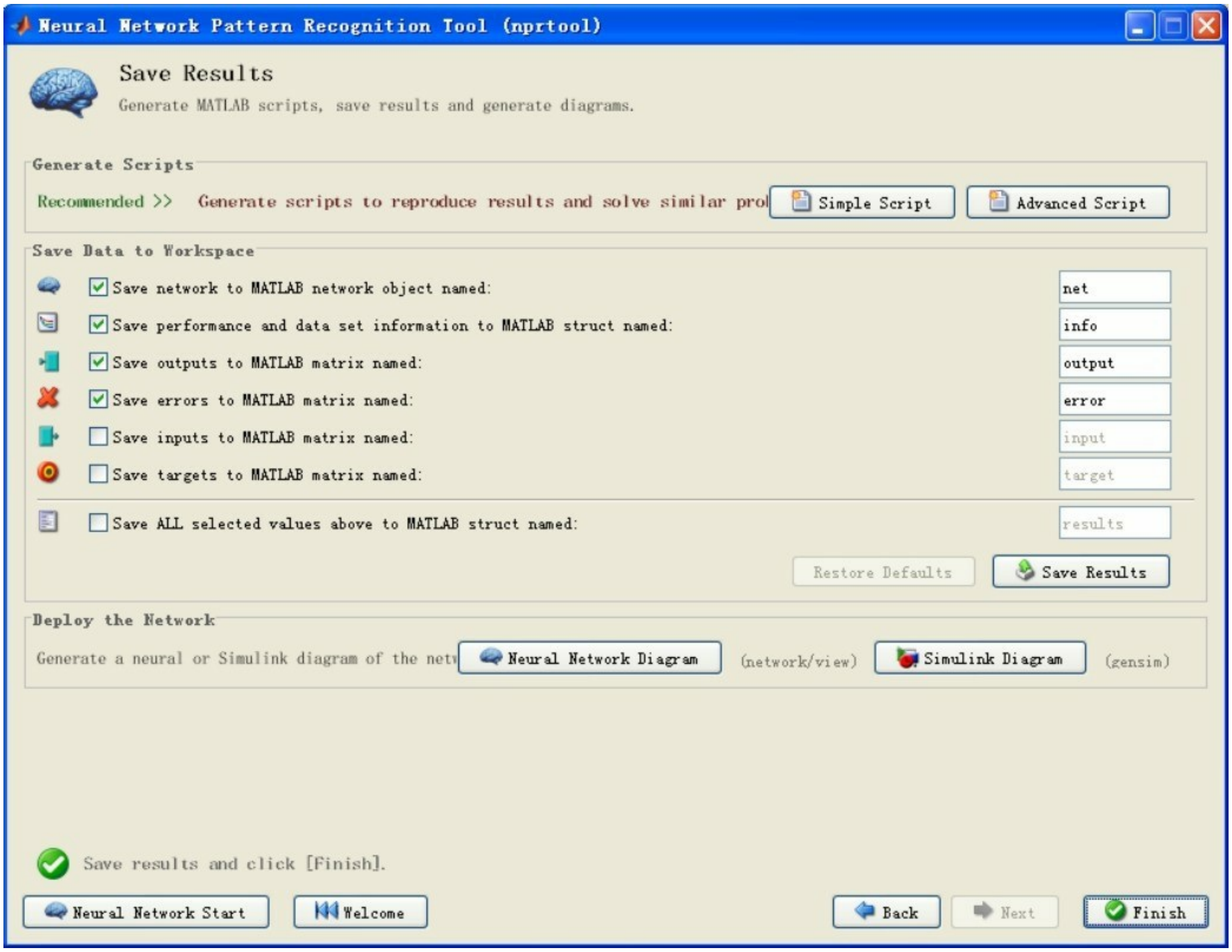


图12-76 保存数据和神经网络的对话框

在MATLAB的Command Window窗口输入以下命令。

nctool

运行后可以得到图12-77所示的数据拟合GUI对话框界面。

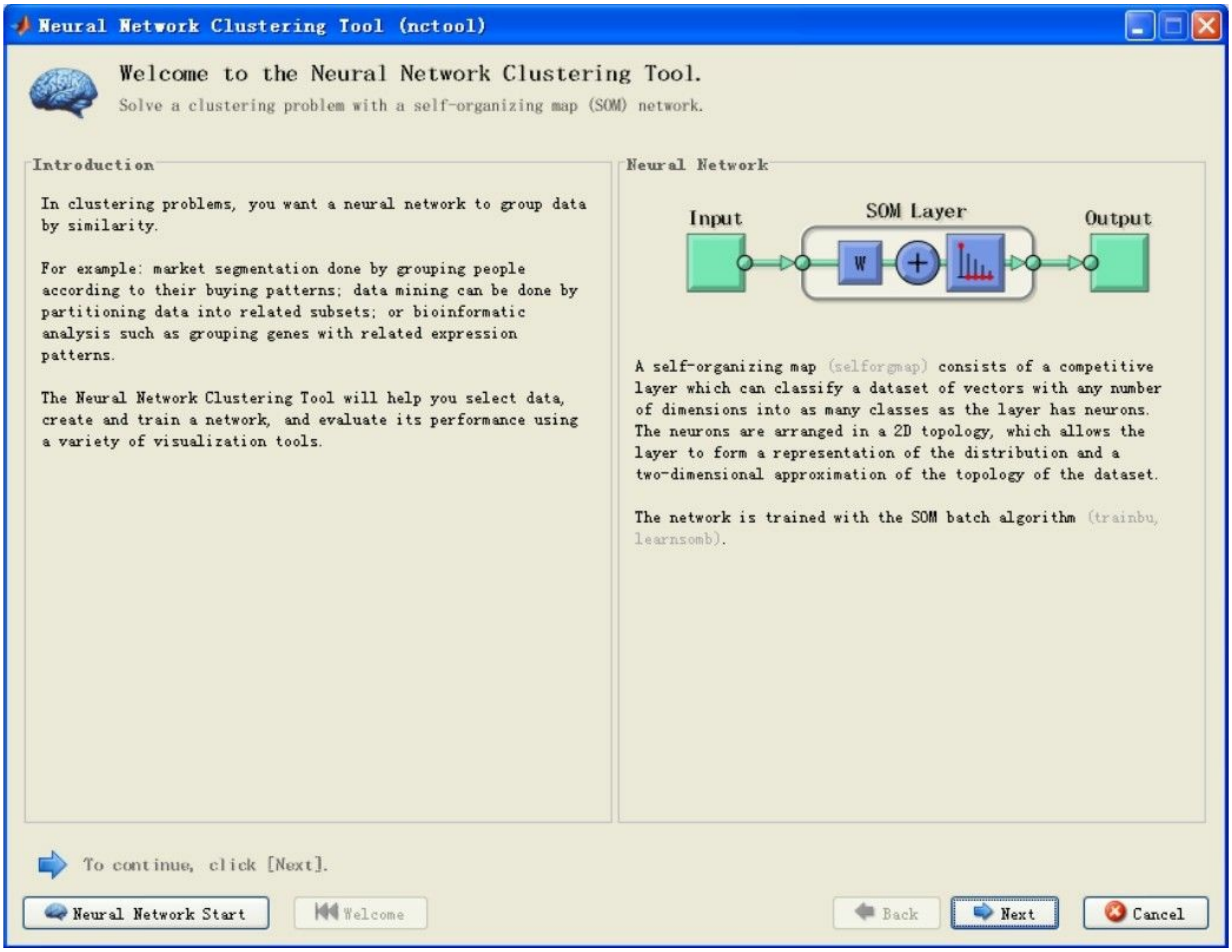


图12-77 数据拟合GUI对话框界面

从图 12-77 中可以看出，神经网络聚类用来收集、建立和训练神经网络，并且利用可视化工具来评价网络的效果。聚类问题往往是建立一种神经网络按照相似性对一组数据进行分组。

MATLAB 使用Self-Organizing Map（自组织特征映射网络，简称SOM）进行数据的聚类。SOM 神经网络一般使用SOM batch 算法，使用的函数是trainubwb和learnsomb 函数。

SOM网络包括一个可以将任意维数的数据分为若干类的竞争层。在竞争层中，神经元按照二维拓扑结构排列，这就使竞争层神经元能够代表与样本分布相似的分布。

单击图12-77中

按钮，得到数据导入对话框。

由于 SOM 网络是无导师、无监督的分类网络，这里不需要输入目标和输出，所以聚类神经网络只需要聚类的数据输入即可。

在MATLAB的Command Window窗口输入以下命令。

x=[1 2 3 4 5];

得到神经网络的输入数据x。

在导入数据的对话框中选择输入数据为x，选择目标数据为t，如图12-78所示。

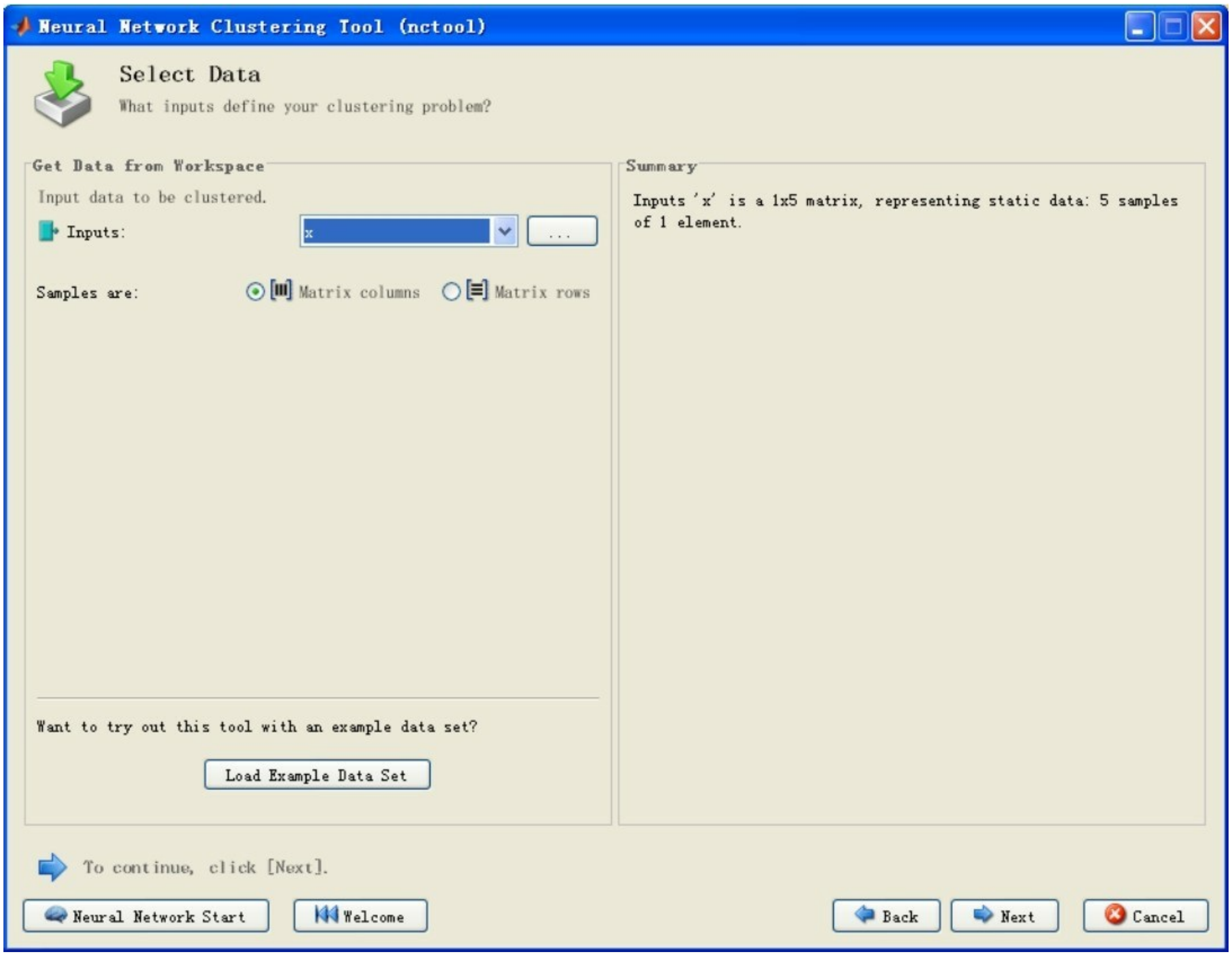
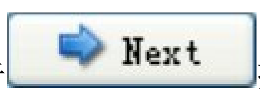


图12-78 选择神经网络的输入数据的对话框

在图12-78中单击

按钮，选取隐含层神经元个数选为15个，得到图12-79所示的选择网络结构对话框。此时，网络的输出有225个。

单击图12-79中的按钮，得到网络训练对话框，如图12-80所示。

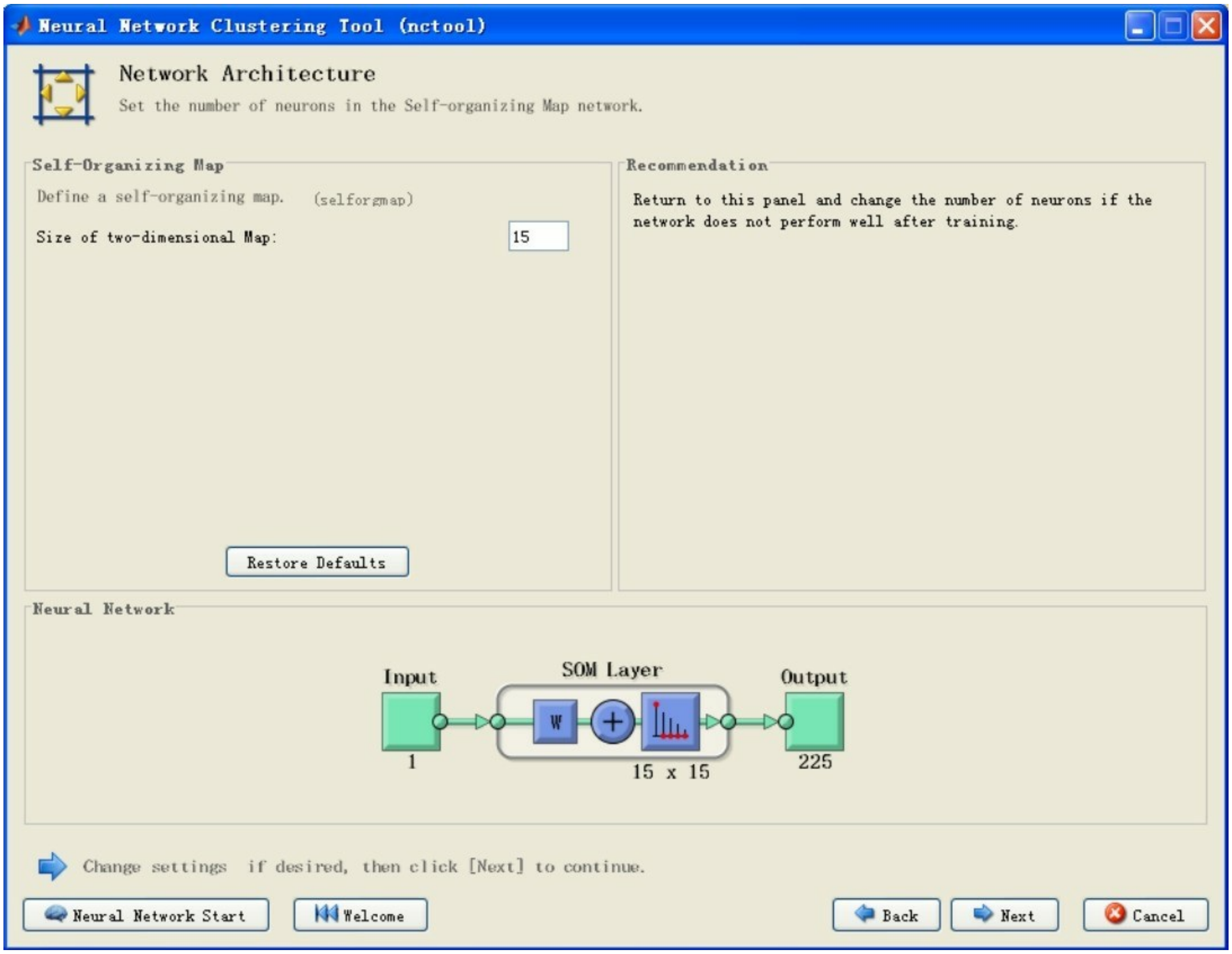


图12-79 选择网络结构对话框

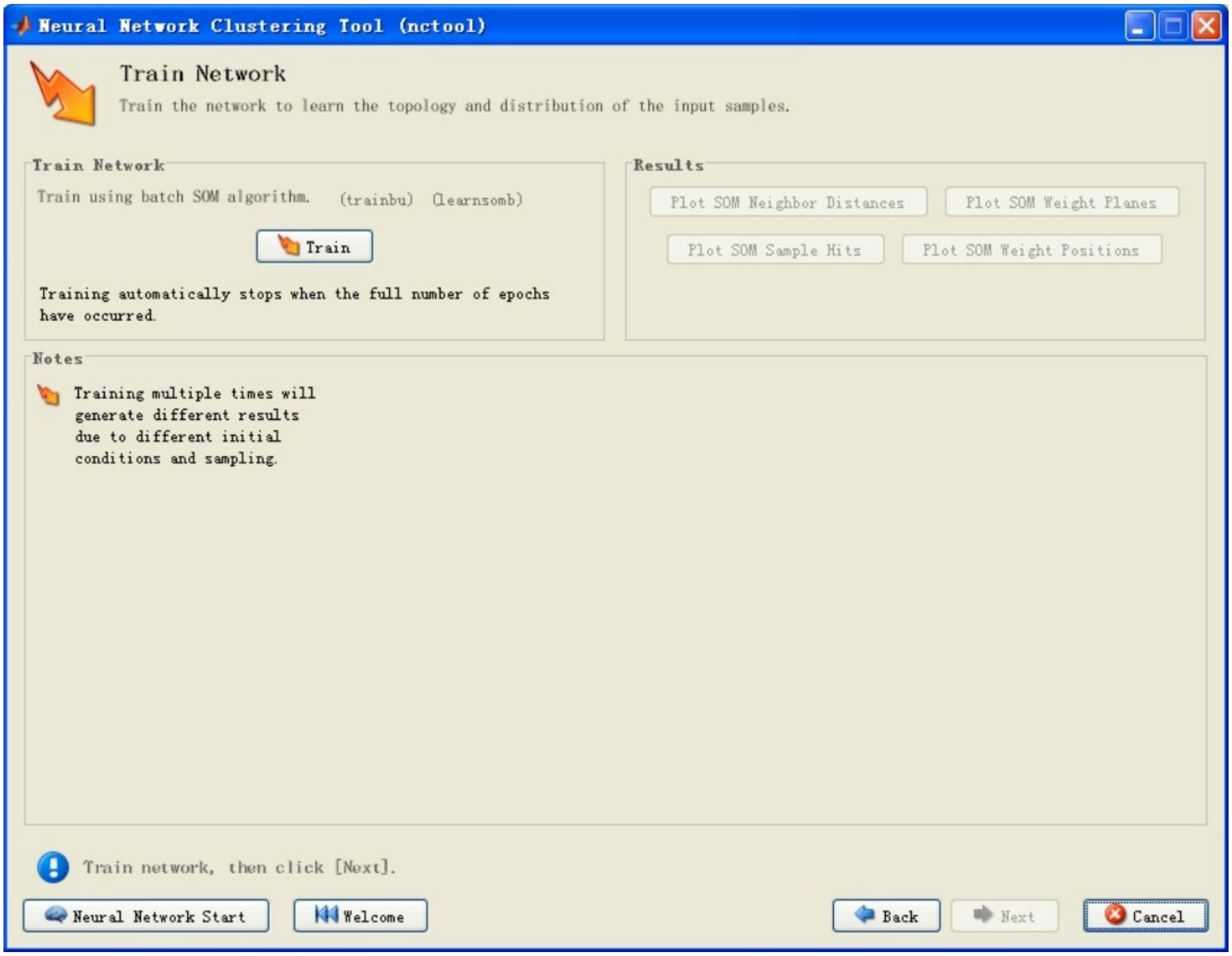


图12-80 网络训练对话框

在图12-80中，单击

按钮，可以对设计的神经网络进行训练。训练后得到的网络训练对话框如图12-81所示。

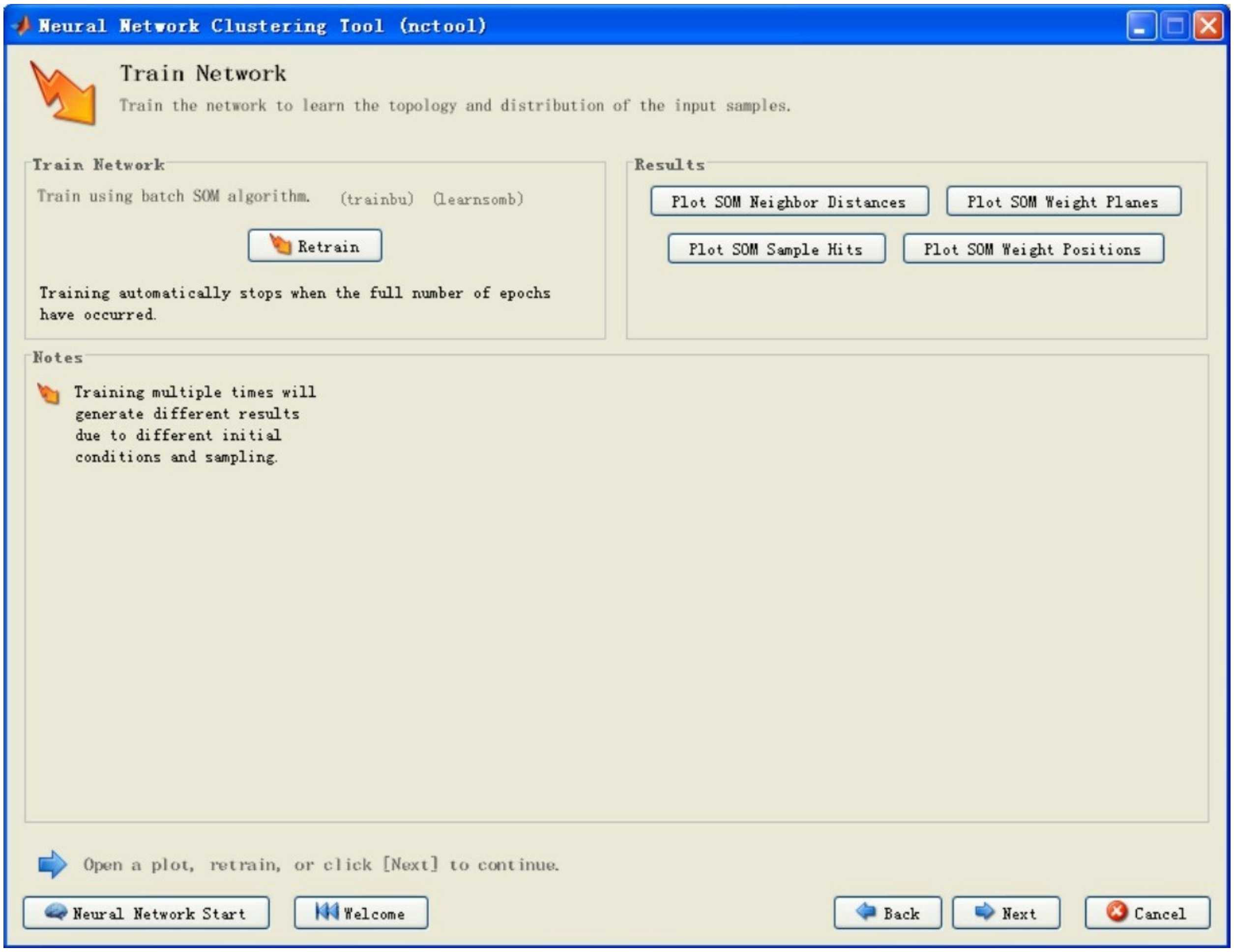


图12-81 神经网络训练后的对话框

从图12-81可以看出，在网络训练结束后，神经网络训练对话框中的 Plot SOM Neighbor Distances 、 Plot SOM Sample Hits 、 Plot SOM Weight Positions 和 Plot SOM Weight Planes 按钮由不可用状态变为可用状态，这两个按钮的作用是查看神经网络聚类的效果。

如单击 Plot SOM Neighbor Distances 按钮，可以得到图12-82所示的临近神经元之间的距离情况。

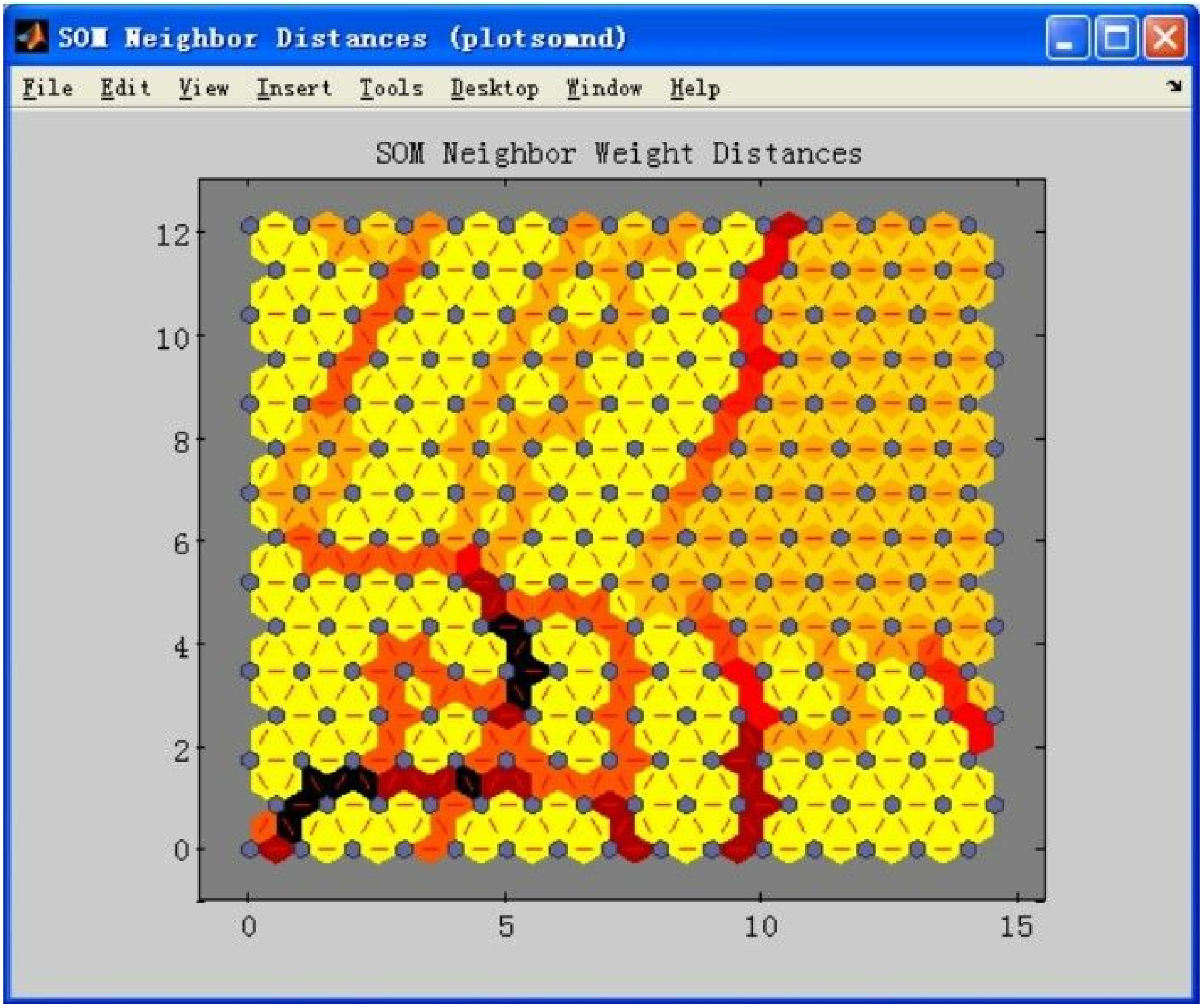


图12-82 临近神经元之间的距离情况

如果单击

按钮，可以得到图12-83所示的每个神经元的分类情况。

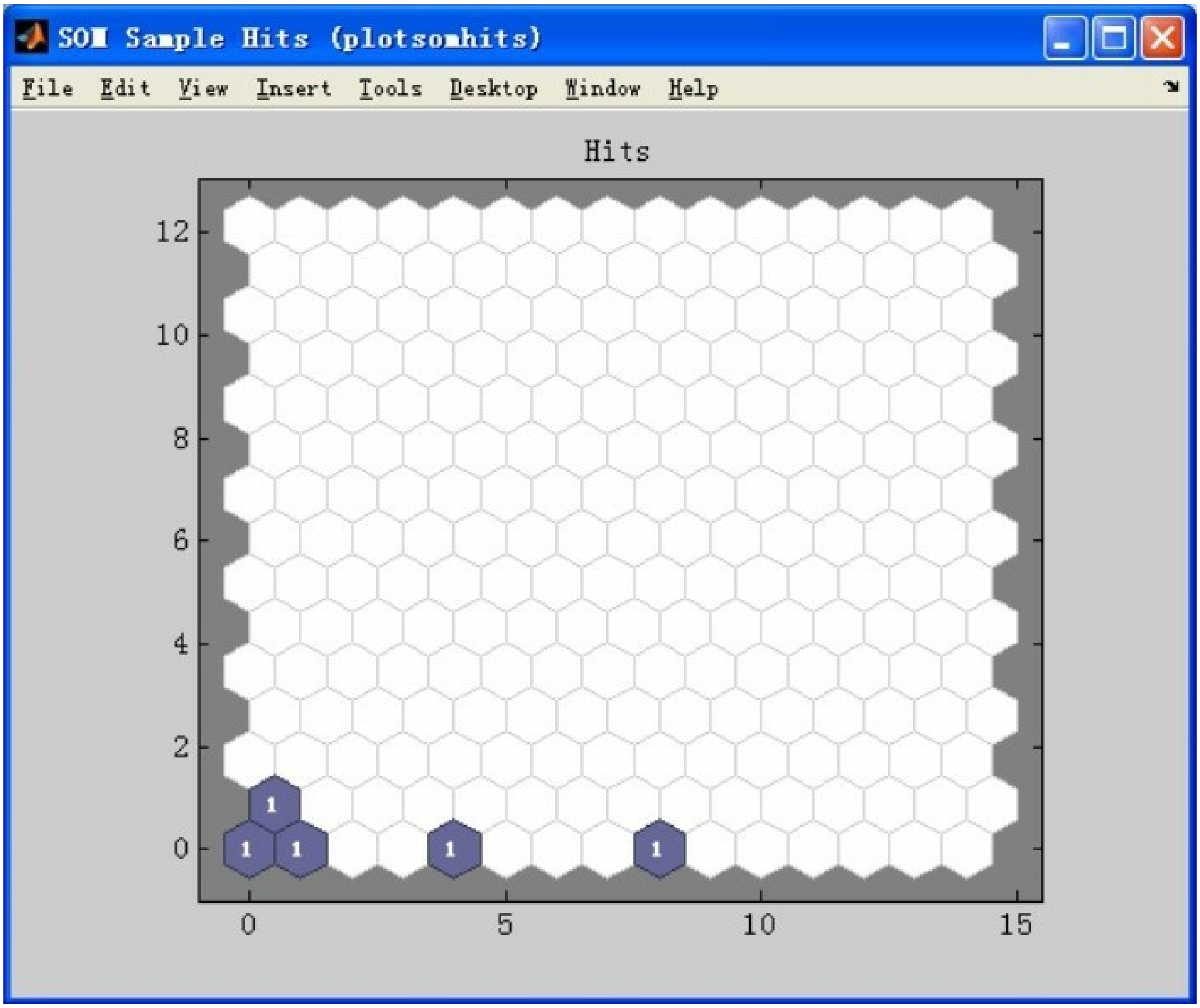
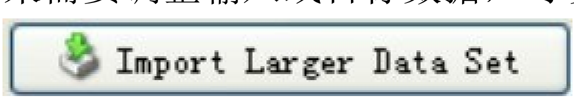


图12-83 每个神经元的分类情况

训练完成后，单击图12-81中的按钮，得到修正神经网络训练的对话框，如图12-84 所示。当神经网络拟合效果不好时，可以单击图中按钮重新训练神经网络；如果需要调整神经网络神经元数目，可以单击按钮；如果需要调整输入或目标数据，可以单击按钮。

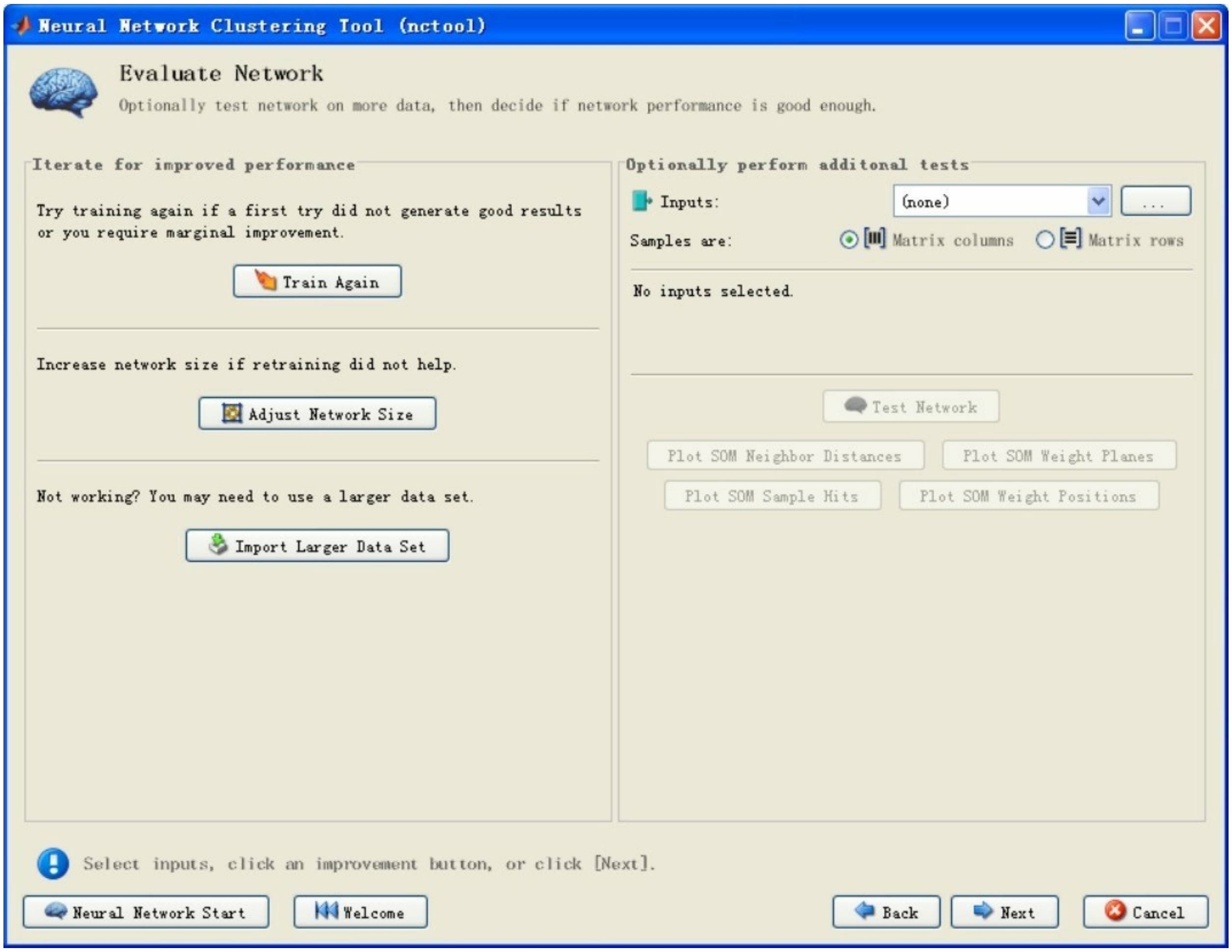


图12-84 修正神经网络训练的对话框

如果需要测试训练好的网络是否达到预期目标，可以设置输入和目标数据，之后图12-84中的 Test Network 按钮被激活，继续单击 Test Network 按钮，Plot SOM Neighbor Distances、Plot SOM Sample Hits、Plot SOM Weight Positions 和 Plot SOM Weight Planes 按钮被激活，如图12-85所示。

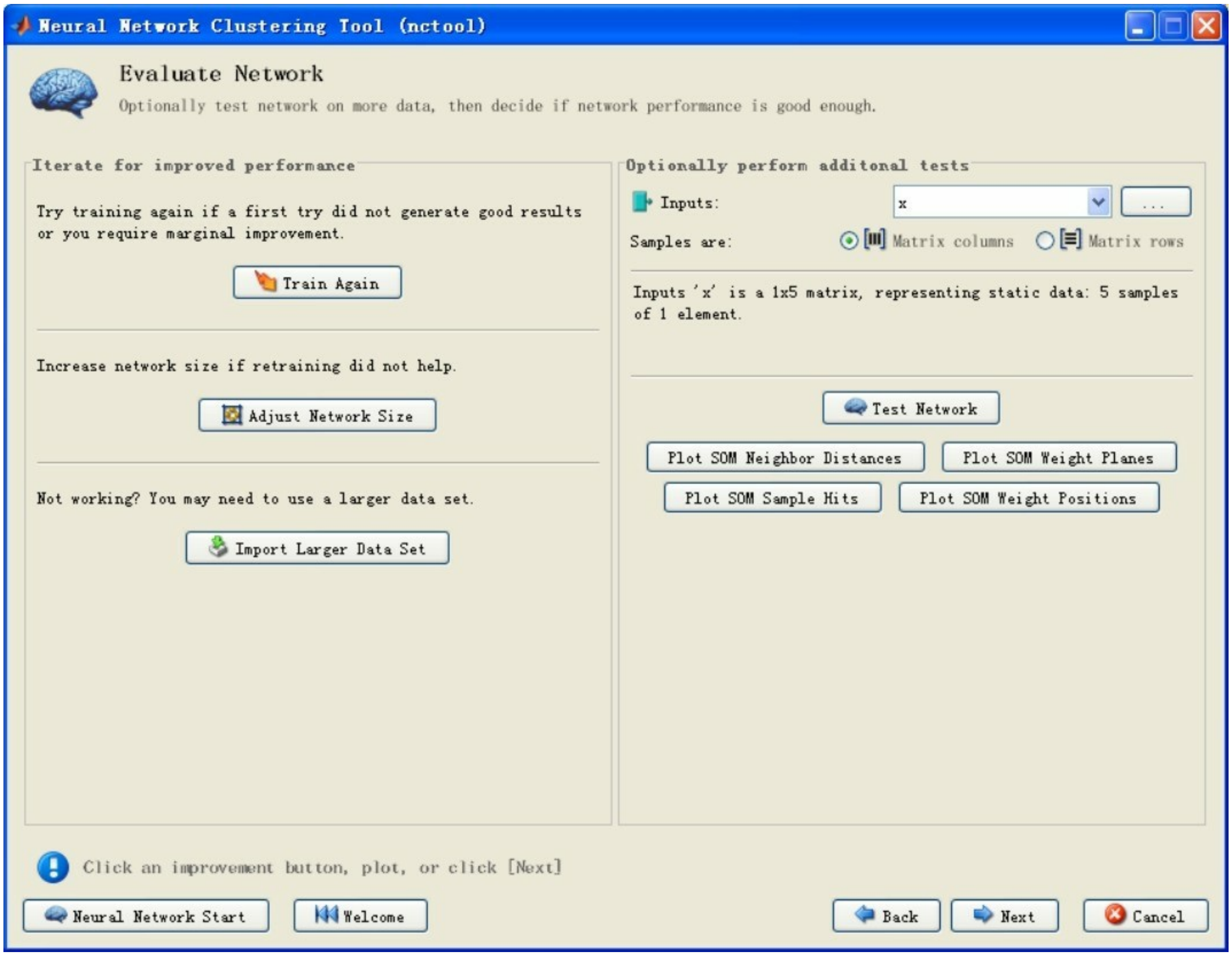


图12-85 Test Network按钮被激活界面

单击

按钮，得到训练好的神经网络测试结果如图12-86所示。图12-86与图12-83的分类情况一致，说明神经网络满足要求。

成功。

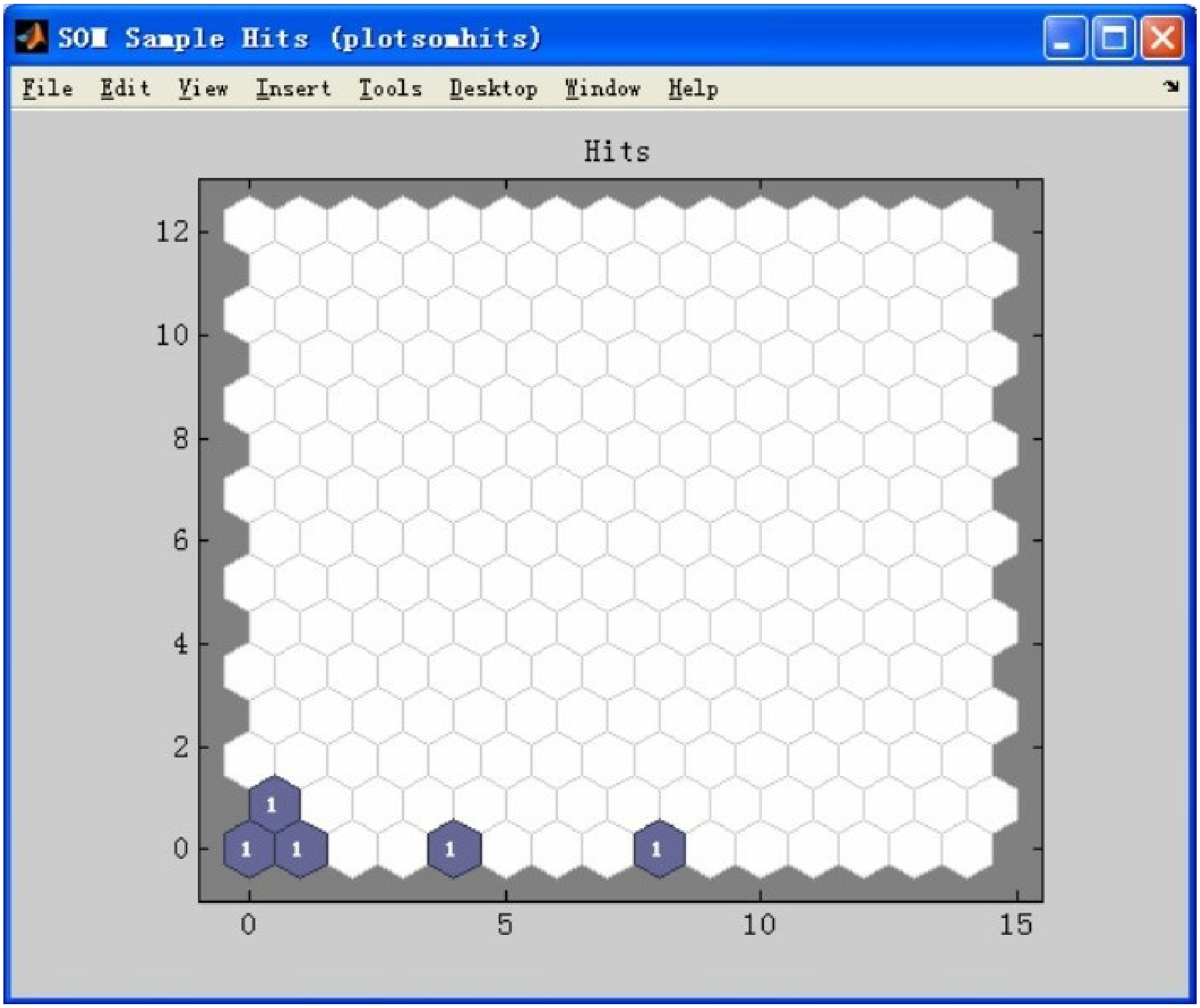
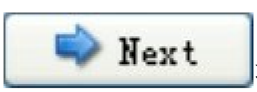


图12-86 每个神经元的分类情况

如果认为网络的拟合效果达到预期目标，单击图12-85中的

按钮，得到最后保存数据和神经网络的对话框，如图12-87所示。

这个对话框运行存储输入、输出、误差和网络结构等与训练相关的数据。

在图 12-87 中单击

按钮， MATLAB 会出现神经网络确认保存警告，单击

按钮，就将训练好的神经网络保存成功。

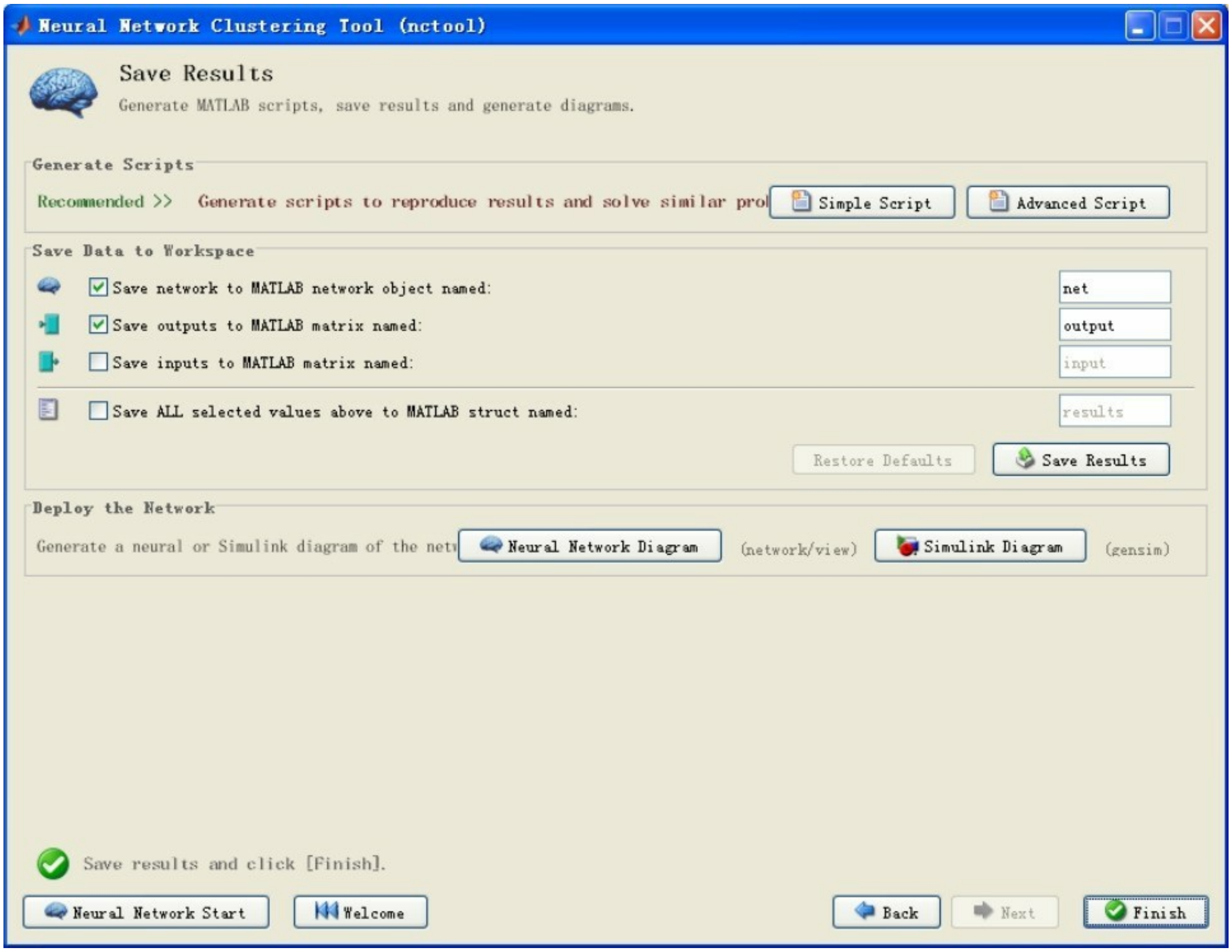


图12-87 保存数据和神经网络的对话框

12.3 GUI数据操作

数据操作一方面指的是将MATLAB中Workspace的数据导入到GUI专供，也可以将GUI中的数据导出到MATLAB的Workspace中；另一方面是指数据的存储、删除、恢复和重新调用等。

12.3.1 从Workspace导入数据到GUI

首先，在MATLAB 的Command Window窗口中输入如下代码。

x=[1 2 3 4 5];

t=[1 0 1 0 1];

这样就得到了数据向量x和t，二者都是由5个元素组成。

在MATLAB的Command Window窗口输入nntool可以打开神经网络GUI 工具窗口，单击

按钮，得到图12-88所示的对话框。

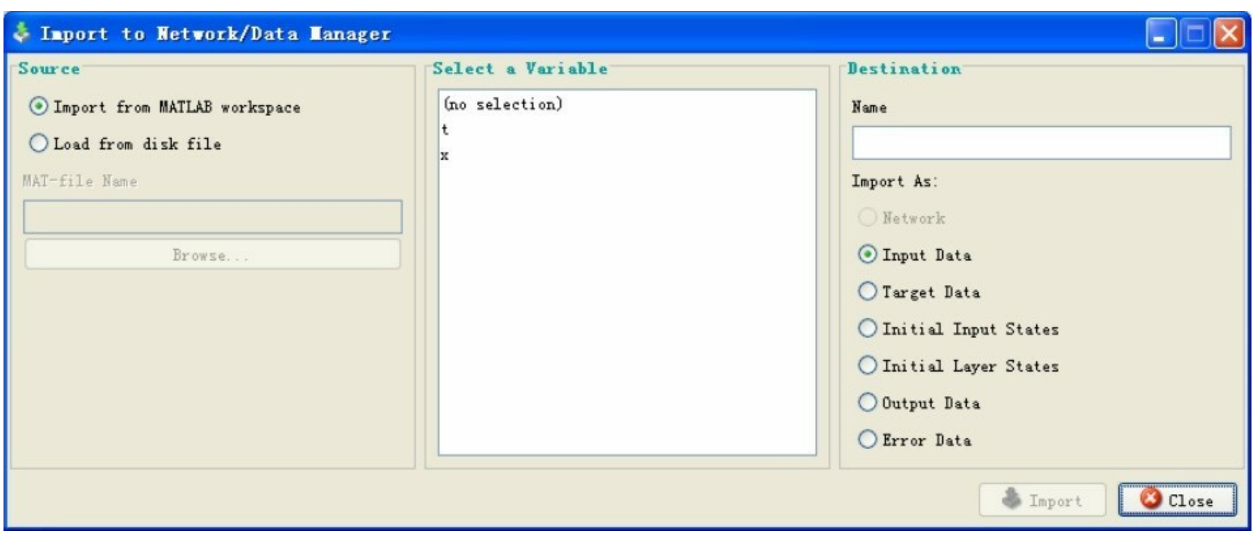


图12-88 数据输入对话框

选中图12-88所示对话框中Source 栏目下方的“Import from MATLAB

workspace”选项，然后在 Select a Variable 栏目下方选中 x 作为神经网络的输入样本数据。同时，在Destination 栏目下方的 Name 选项中输入神经网络输入样本数据的名称。最后选中“Input Data”选项，表示该数据是神经网络的输入数据，神经网络输入数据的设置如图12-89所示。

提示：在Name 选项中，一般默认选择Select a Variable中的变量名称。

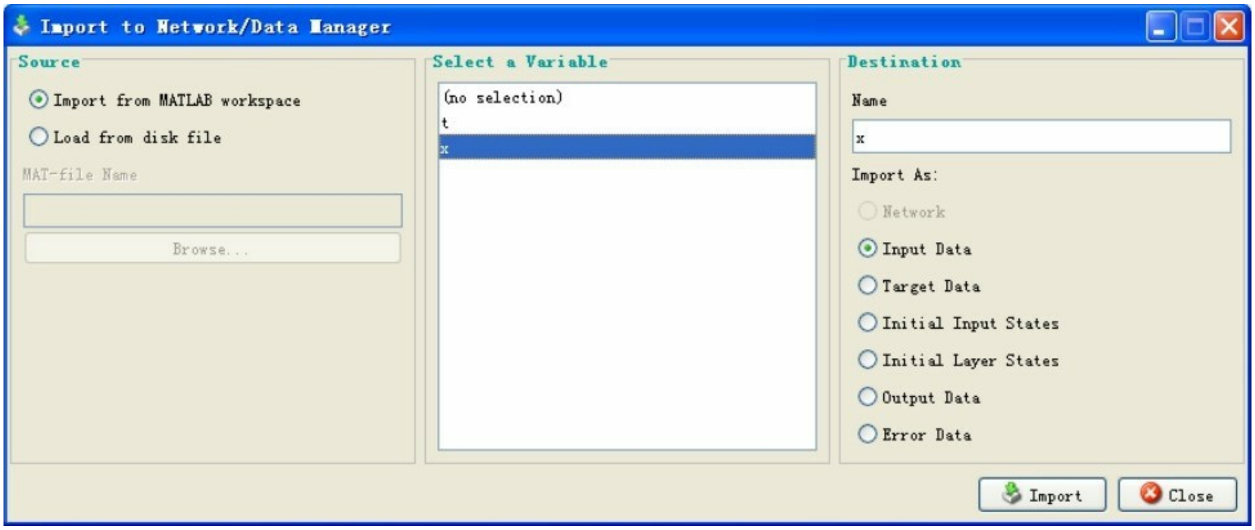


图12-89 神经网络输入数据的设置

在图12-89中，单击按钮，MATLAB会出现图12-90所示的提示框，该提示框表明输入变量x生成成功。

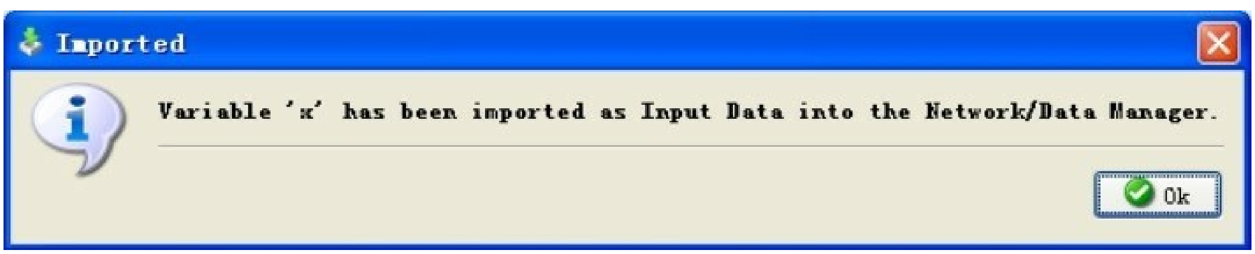


图12-90 生成变量提示框

此时，神经网络GUI 工具窗口的Input Data 下方显示出神经网络的输入变量x，如图12-91所示。

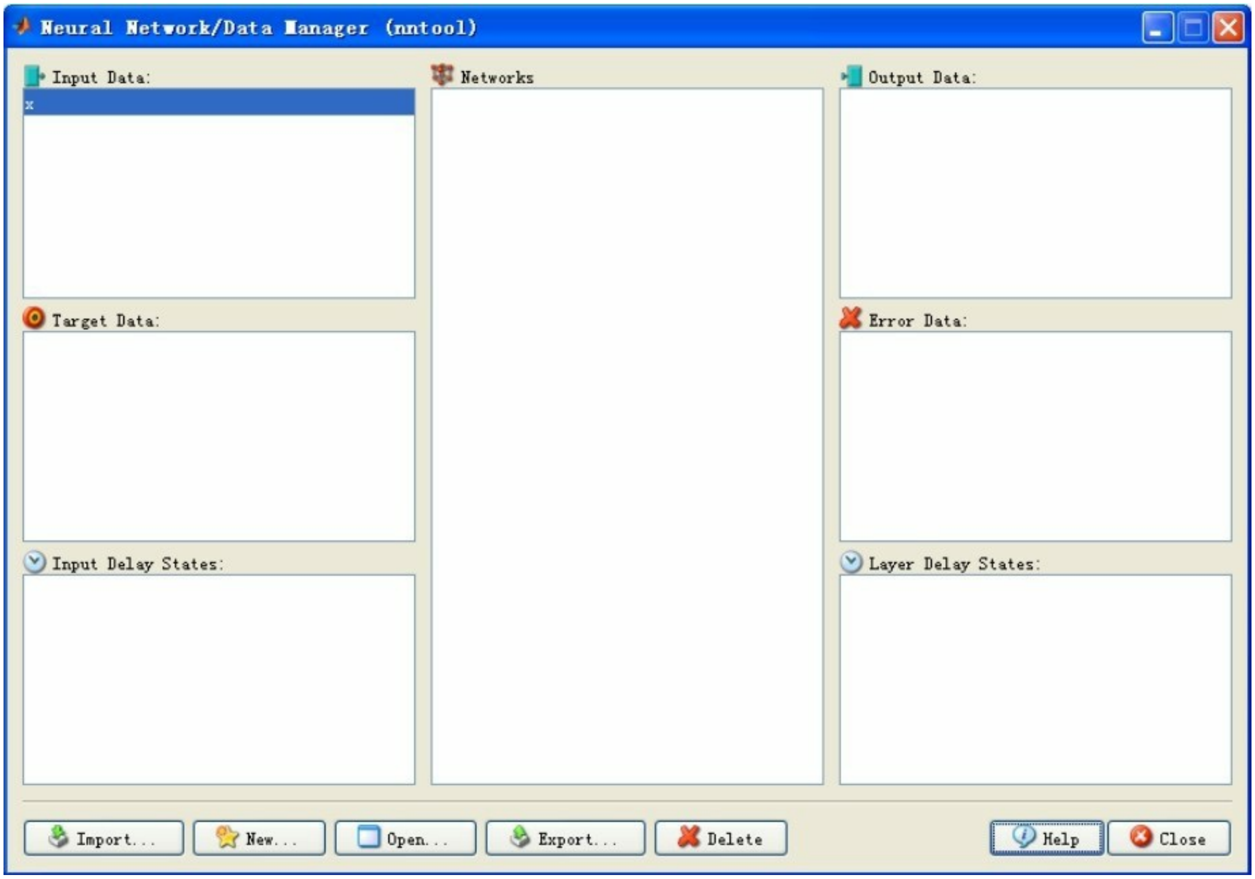


图12-91 生成输入变量后的神经网络GUI工具窗口

按照上述方法，还可以设置神经网络的目标向量、初始输入状态向量等。

此外，在图12-89中还有一个“Load from disk file”选项，该选项表示从磁盘中导入数据。

12.3.2 从GUI中导出数据到Workspace

通过 GUI 得到的网络仿真结果、训练结果和误差等，都可以导出到 MATLAB 的Workspace中。在MATLAB的Command Window中可以利用这些数据进行一些其他的操作。

按照12.2.1小节中的方法，建立一个神经网络，并完成训练和仿真，得到数据如图12-92所示。

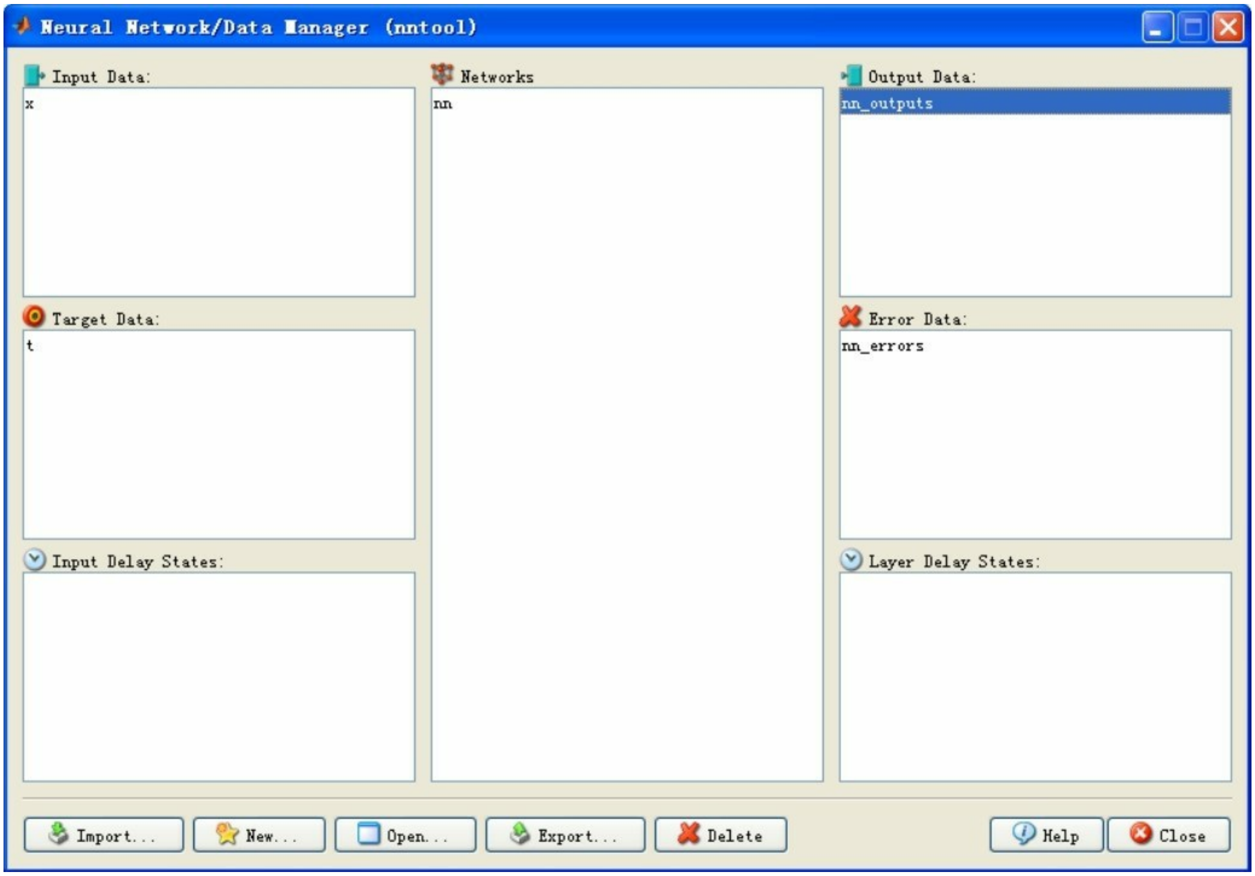


图12-92 训练和仿真网络得到的数据

在图12-92 中，选择Output Data 选项下的nn_outputs 变量后，单击

按钮，得到图12-93所示导出变量对话框。

当选中图12-93中的数据nn_outputs后，按钮

被激活，如图12-94所示。

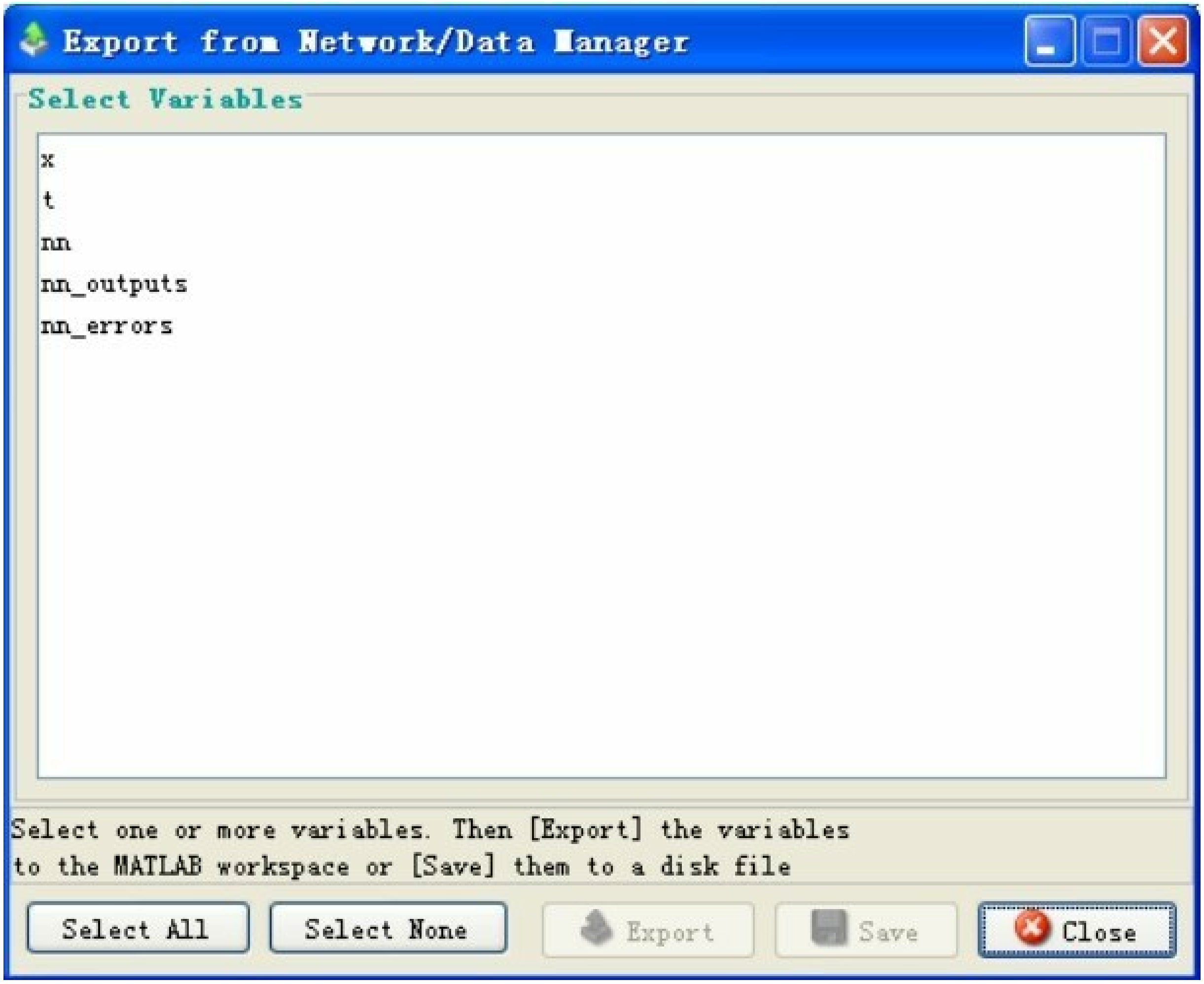


图12-93 导出变量对话框

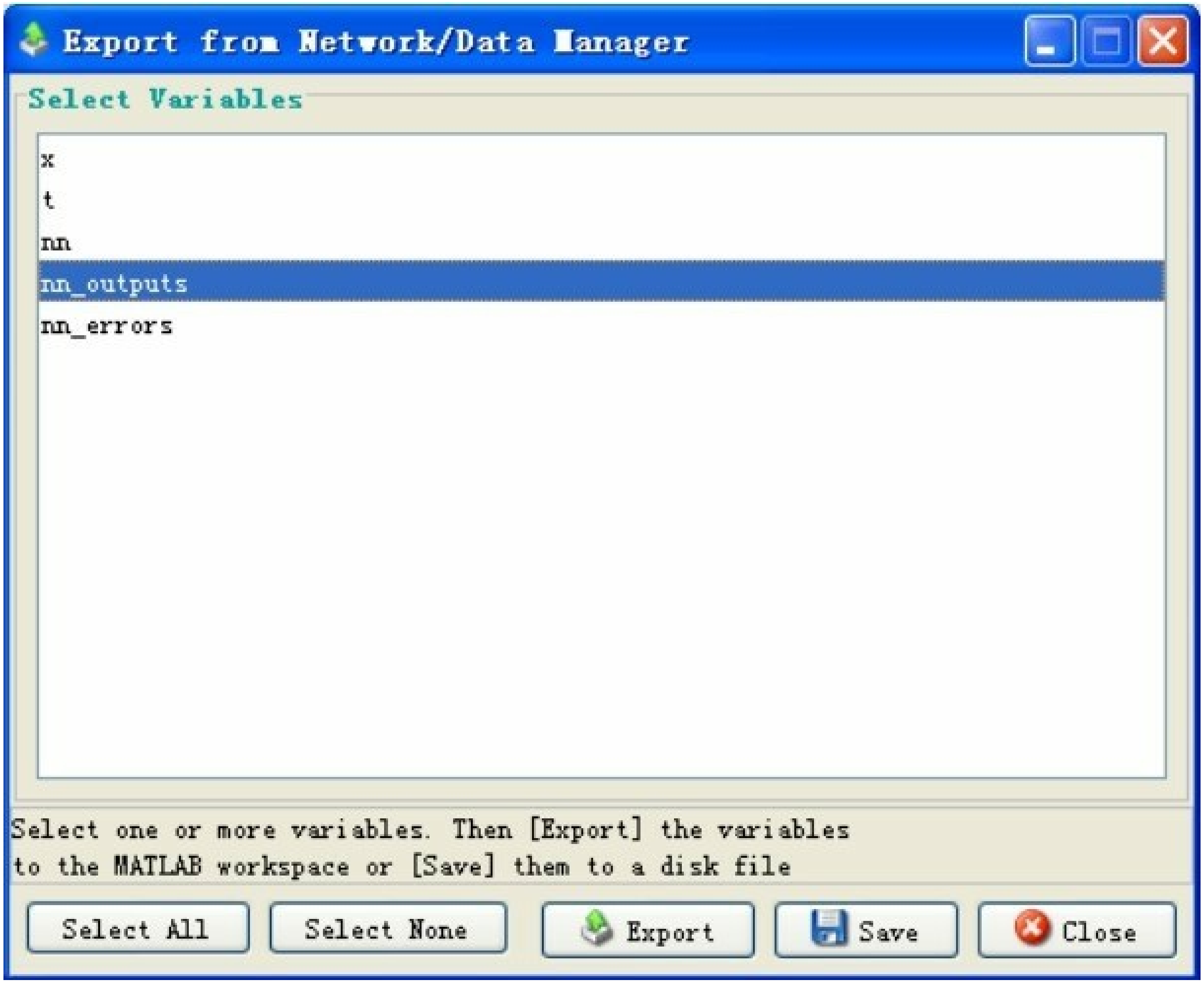


图12-94 激活Export按钮对话框

单击

按钮，MATLAB显示图12-95所示的变量导出成功的提示框。

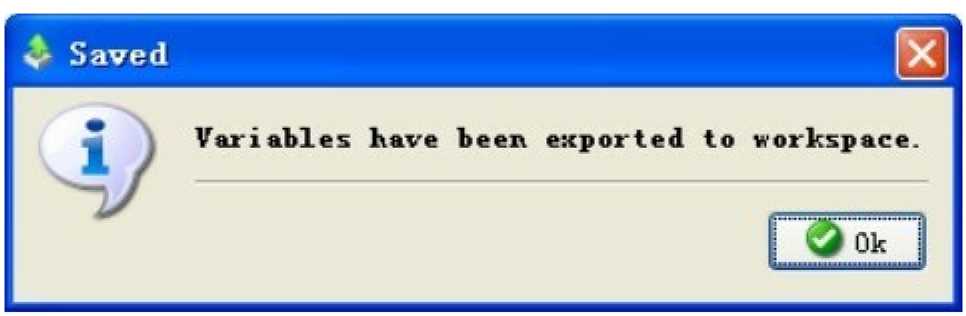


图12-95 变量导出成功提示框

如果要确认以上过程是否成功，可以在MATLAB的Command Window中输入命令who来检测此时工作空间中的所有变量，得到以下结

果。

>> who

Your variables are:

nn_outputs   t   x

这表明nn_outputs已经成功导入到MATLAB的Workspace中。

按照上述方法可以将神经网络也导入到MATLAB的Workspace中。

如果需要查看神经网络的基本信息，可以在MATLAB 的Command Window中输入建立神经网络的名称nn，得到如下所示信息。

```
>> nn
nn =
  Neural Network
    name: 'Custom Neural Network'
    efficiency: .cacheDelayedInputs, .flattenTime,.memoryReduction,
.flattenedTime
    userdata: (your custom info)
  dimensions:
    numInputs: 1
    numLayers: 2
    numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
    numFeedbackDelays: 0
    numWeightElements: 31
    sampleTime: 1
  connections:
    biasConnect: [1; 1]
    inputConnect: [1; 0]
```

layerConnect: [0 0; 1 0]

outputConnect: [0 1]

subobjects:

inputs:{1x1 cell array of 1 input}

layers:{2x1 cell array of 2 layers}

outputs:{1x2 cell array of 1 output}

biases:{2x1 cell array of 2 biases}

inputWeights:{2x1 cell array of 1 weight}

layerWeights:{2x2 cell array of 1 weight}

functions:

adaptFcn: 'adaptwb'

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: 'dividerand'

divideParam: .trainRatio, .valRatio, .testRatio

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mse'

performParam: .regularization, .normalization

plotFcns:{'plotperform', plottrainstate,plotregression}

plotParams:{1x3 cell array of 3 params}

trainFcn: 'trainlm'

trainParam: .showWindow, .showCommandLine, .show, .epochs, .time, .goal, .min_grad, .max_fail, .mu, .mu_dec, .mu_inc, .mu_max

weight and bias values:

IW:{2x1 cell} containing 1 input weight matrix

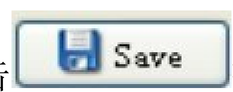
LW:{2x2 cell} containing 1 layer weight matrix

b:{2x1 cell} containing 2 bias vectors
methods:
adapt: Learn while in continuous use
configure: Configure inputs & outputs
gensim: Generate Simulink model
init: Initialize weights & biases
perform: Calculate performance
sim: Evaluate network outputs given inputs
train: Train network with examples
view: View diagram
unconfigure: Unconfigure inputs & outputs
evaluate: outputs=nn(inputs)

12.3.3 数据的存储和读取

通过GUI定义的数据、网络等信息可以存储到用户指定的位置，需要的时候再加载，这样就可以使得设计好的网络重复使用，节省网络的设计和训练时间。

以图12-94中所示的数据nn_outputs存储和读取为例，介绍数据的存储和读取过程。

选择图12-94中的数据nn_outputs后，单击

按钮，出现Save to a MAT file 对话框，如图12-96所示。

在图12-96中选择数据需要保存的路径后，单击

按钮，MATLAB出现图12-97所示的数据保存成功提示框。

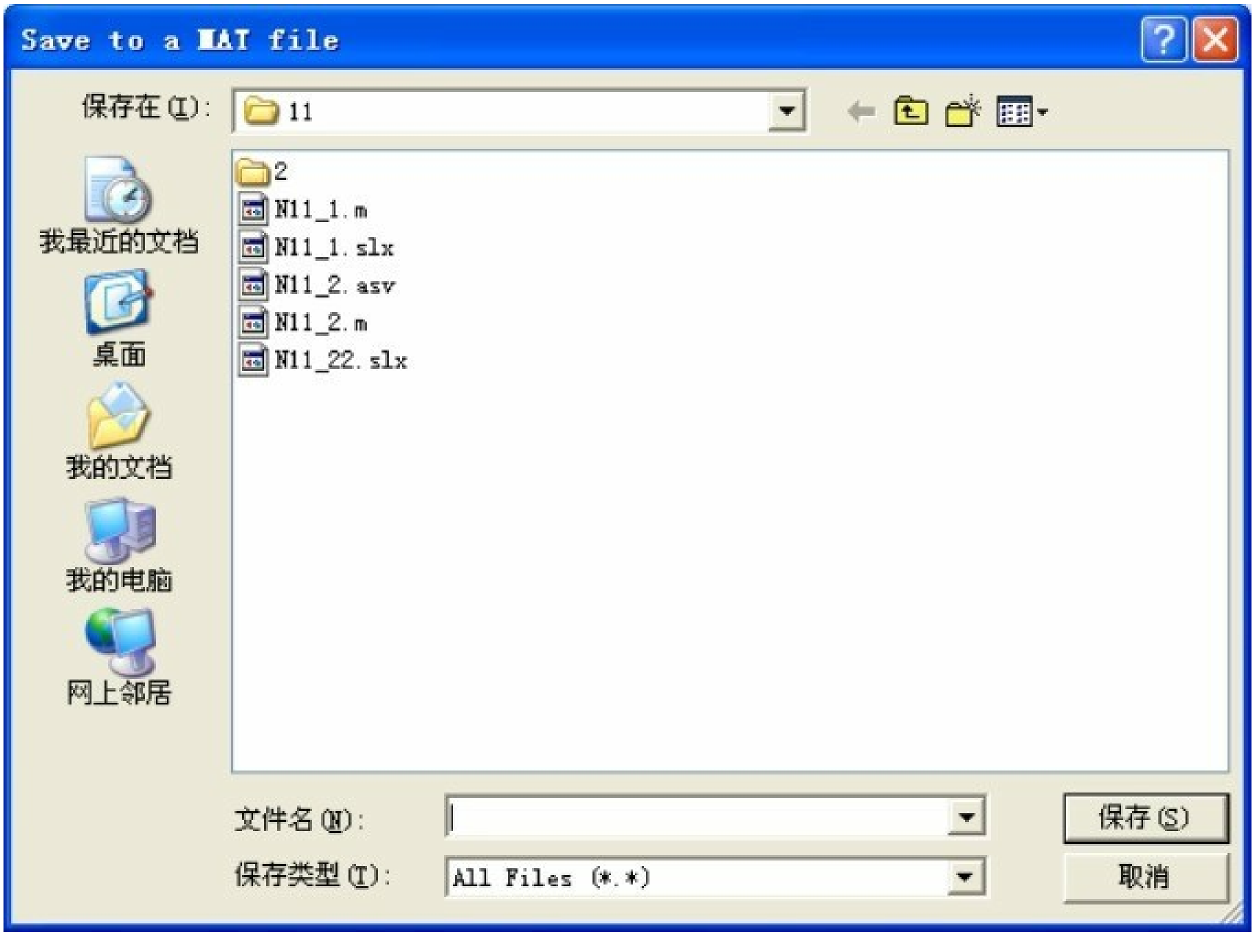


图12-96 Save to a MAT file对话框



图12-97 数据保存成功提示框

如果需要读取图12-97提示保存成功的数据N11_2213，可以选择图12-92中的按钮，在出现的Import to Network/Data Manager窗口中，选择“Load from disk file”选项，如图12-98所示。

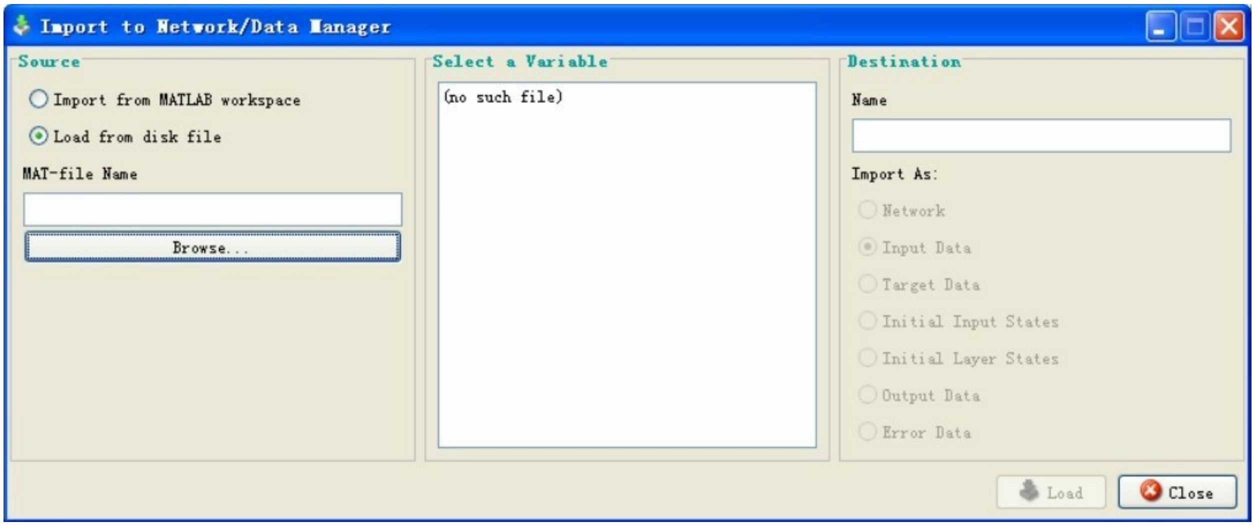
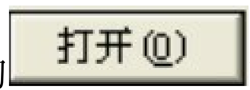


图12-98 Import to Network/Data Manager窗口

单击图12-98中的

按钮，在出现的Select MAT file窗口中选择数据N11_2213所在的路径，并在该路径中选择数据N11_2213，如图12-99所示。

单击图12-99中的

按钮后，得到Import to Network/Data Manager窗口。在此窗口中，选择需要读取的数据nn_outputs，并在Destination 下方选择“Input Data”选项，如图12-100所示。

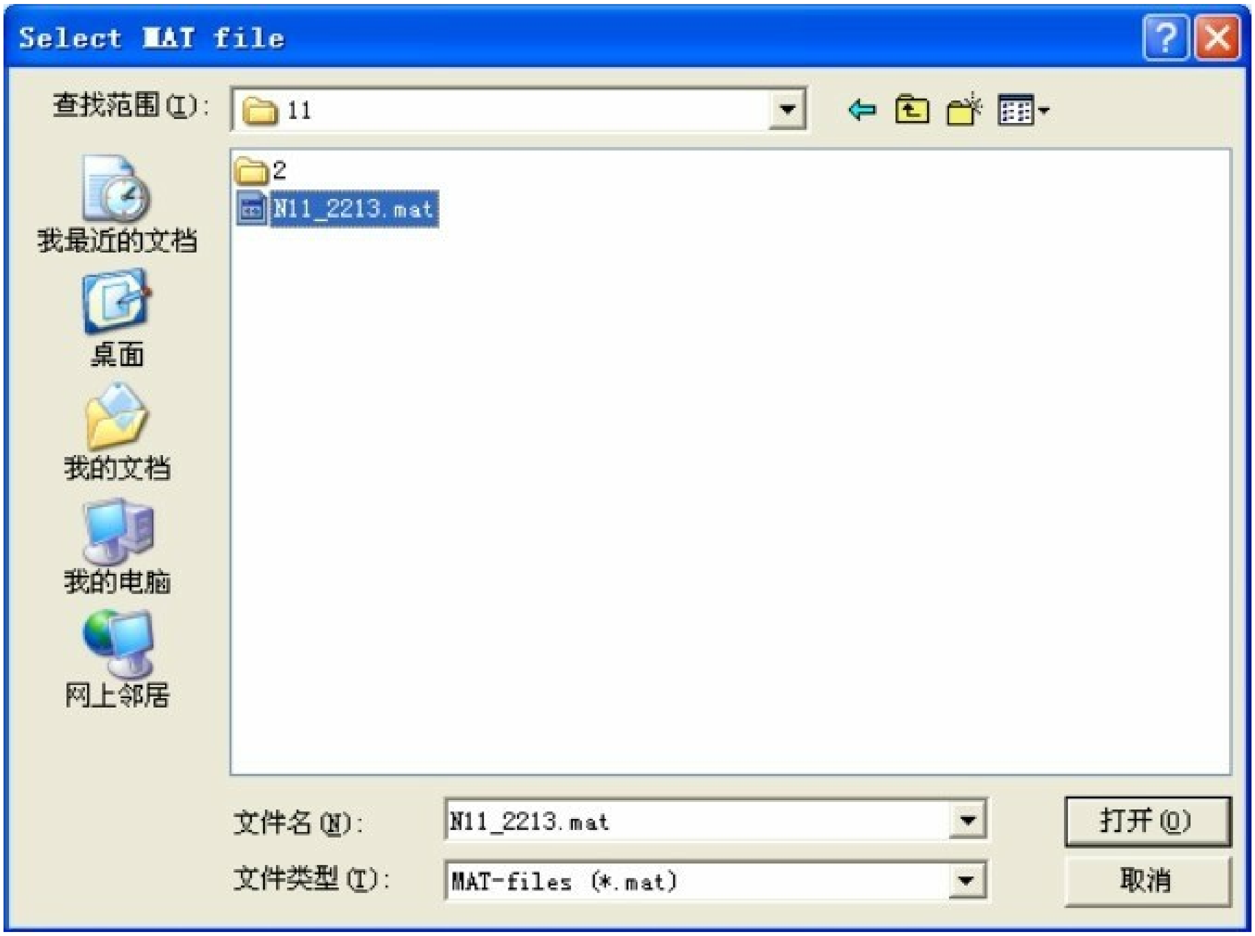


图12-99 Select MAT file窗口

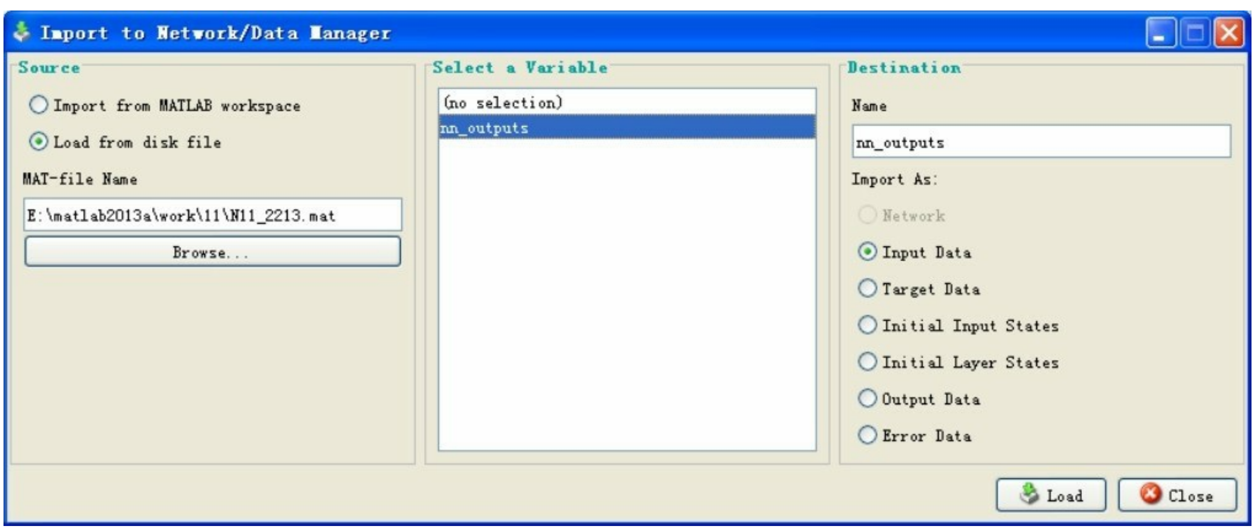
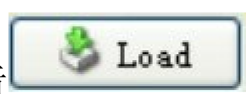


图12-100 选择需要读取的数据nn_outputs

在图12-100中，单击按钮，则完成数据nn_outputs的

读取。从图12-101中可以看出，数据nn_outputs 被读取为Input Data。

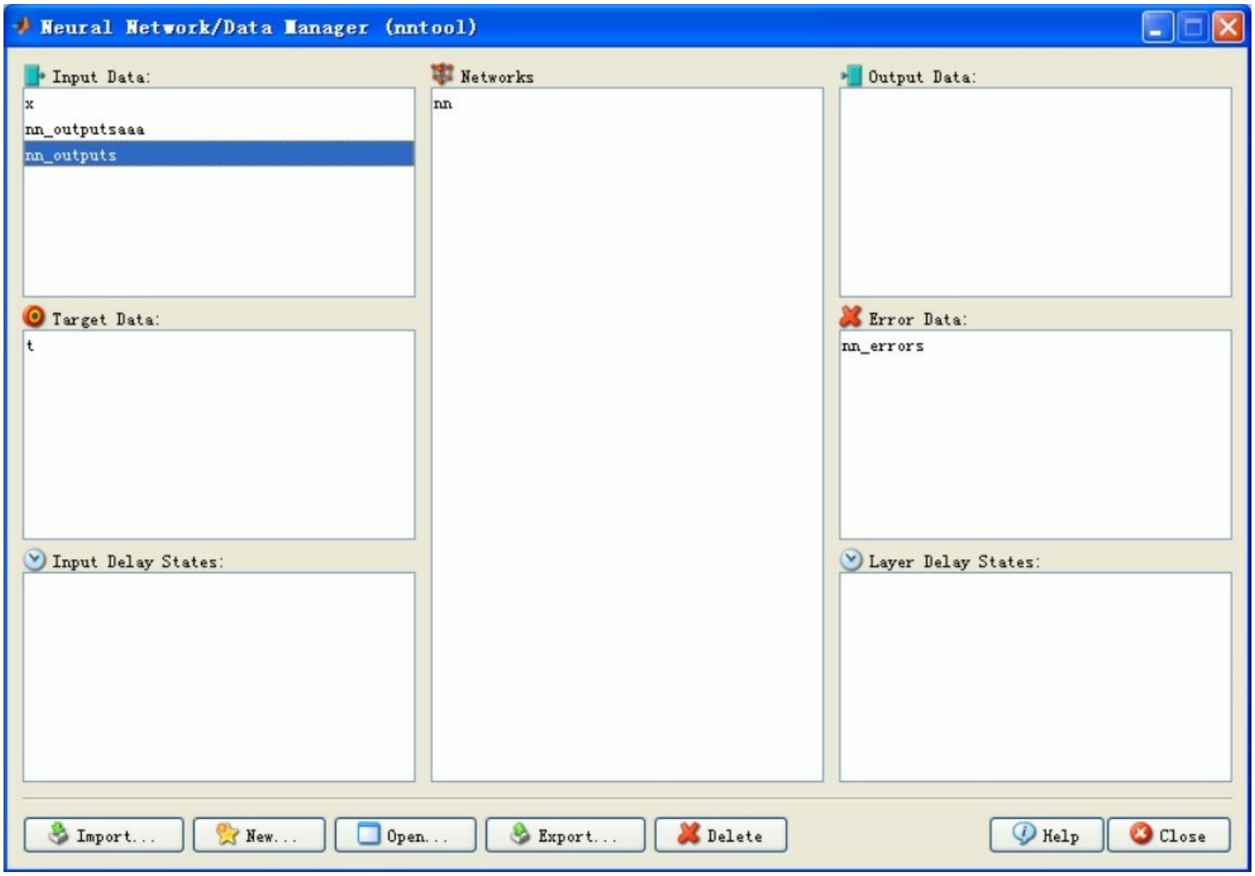


图12-101 读取nn_outputs 数据到Input Data中

12.3.4 数据的删除

在通过GUI进行网络设计时，有时候会出现误操作，如输入数据出现错误。此时为了设计新的输入数据，需要删除原有的错误输入数据。

数据删除最简单的方法是强行关闭 MATLAB 窗口，在提示的数据保存提示框中选择No。但这种方法很容易将需要保存的数据也删除。

最好的方法是在 Neural Network/Data Manager 窗口中选择要删除的数据，然后单击Neural Network/Data Manager窗口中的

按钮，这样就可以删除出现错误或多余的数据了。

在图12-101中，单击按钮，得到图12-102 所示Neural Network/Data Manager窗口。相对于图12-101，图12-102中已经将数据nn_outputs成功删除。

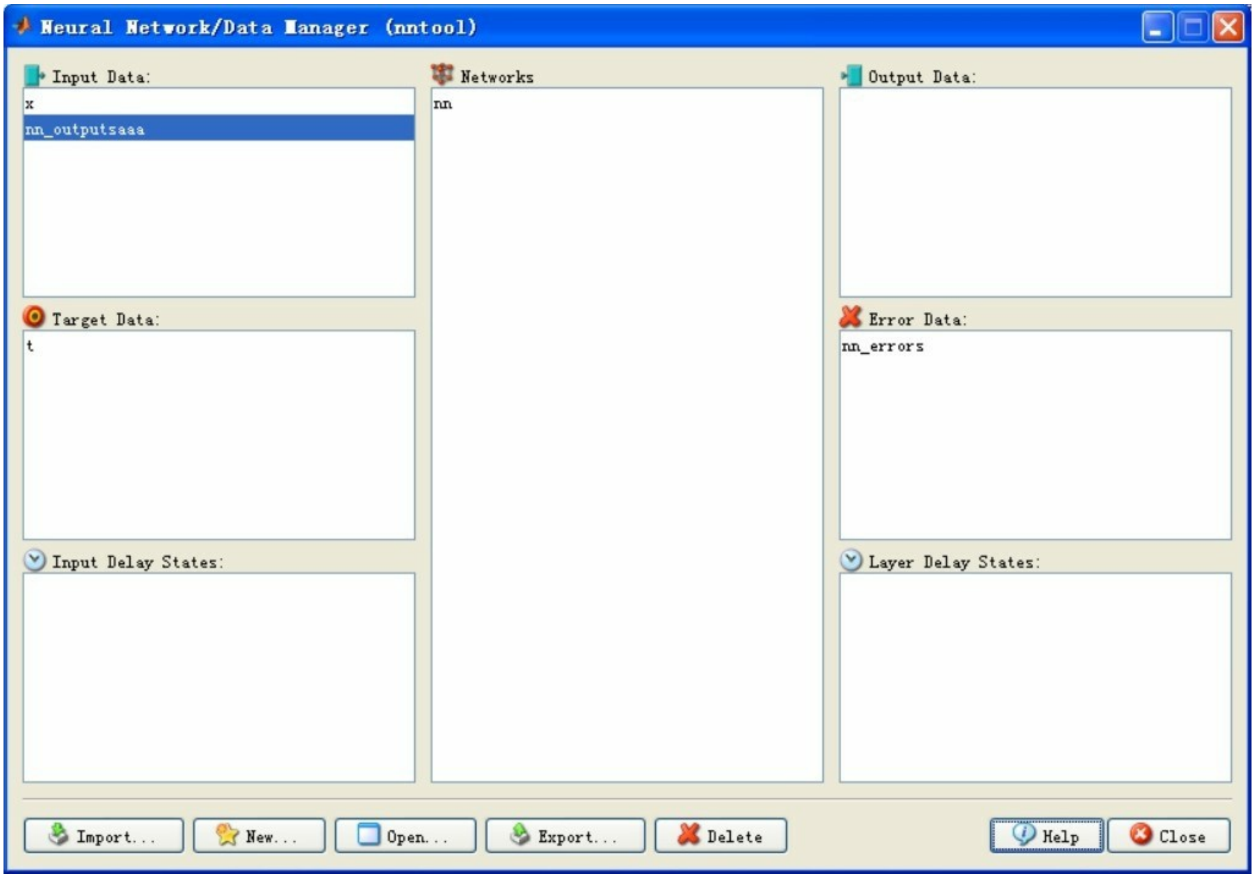


图12-102 成功删除数据nn_outputs后的Neural Network/Data Manager窗口

12.4 本章小结

GUI 是 Simulink 的重要功能，其使得 Simulink 模型更加直观。本章首先介绍了 GUI的基本概念，然后重点对神经网络GUI的几种模型进行了详细介绍，最后介绍了如何使用GUI对数据进行操作。

第13章 自定义神经网络及函数

一般来说，MATLAB软件中自带的网络模型已经足够用户使用。但如果用户需要开发更加复杂的神经网络模型，现有软件中的标准函数就不能满足用户的需求了。此时，神经网络工具箱给用户提供了另外一种方法，用户可以根据自己的需要自定义神经网络。

本章将重点介绍自定义神经网络和自定义函数。

学习目标：

·掌握自定义神经网络的方法

·熟悉自定义函数的方法

13.1 自定义神经网络

假设有一个如图13-1所示的自定义神经网络，输入向量均为输入序列向量。

p_1={[0;0] [2;0.5]};

p_2={[2;-2;1;0;1] [-1;-1;1;0;1]};

希望从第2个网络层得到对p_1，p_2的延迟响应输出（y^1）；从第3个网络层得到目标序列响应输出（y^2），其目标序列向量如下。

T={1, -1};

由于神经网络工具箱没有提供创建该网络的函数，所以需要自定义神经网络。

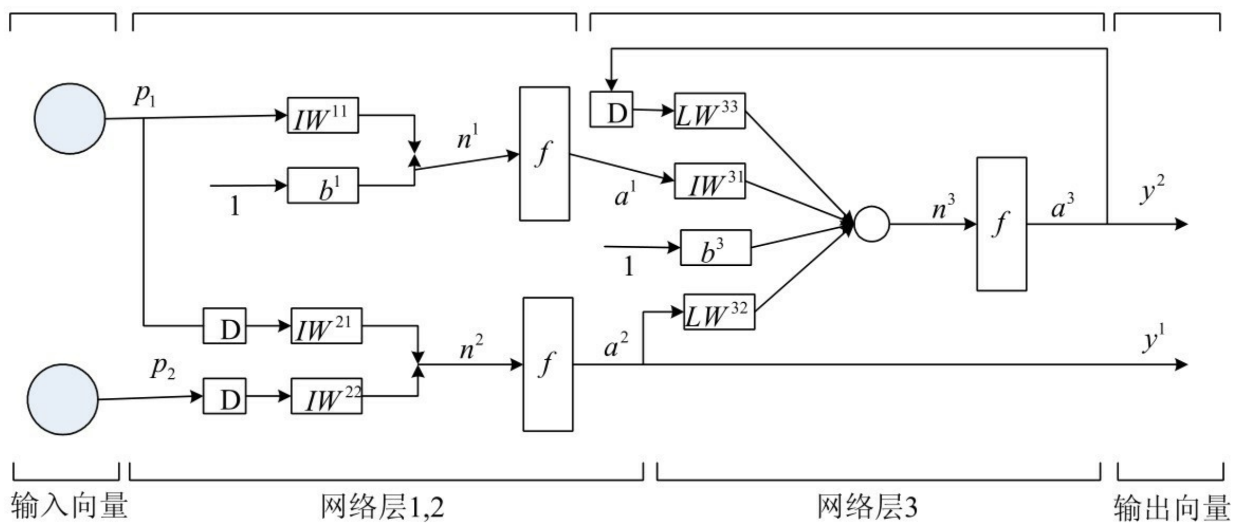


图13-1 自定义神经网络结构图

下面以此为例，介绍创建自定义神经网络的方法。

13.1.1 网络的创建

首先在MATLAB 的Command Window窗口输入下列命令。

>> net=network

生成自定义神经网络的结构。该命令运行的结果如下所示。

```
net =
    Neural Network
        name: 'Custom Neural Network'
        efficiency: .cacheDelayedInputs, .flattenTime,
        .memoryReduction
        userdata: (your custom info)
    dimensions:
        numInputs: 0
        numLayers: 0
        numOutputs: 0
        numInputDelays: 0
        numLayerDelays: 0
        numFeedbackDelays: 0
        numWeightElements: 0
        sampleTime: 1
    connections:
        biasConnect: []
        inputConnect: []
        layerConnect: []
        outputConnect: []
    subobjects:
        inputs:{0x1 cell array of 0 inputs}
        layers:{0x1 cell array of 0 layers}
        outputs:{1x0 cell array of 0 outputs}
        biases:{0x1 cell array of 0 biases}
```

```
    inputWeights:{0x0 cell array of 0 weights}
    layerWeights:{0x0 cell array of 0 weights}
functions:
    adaptFcn: (none)
    adaptParam: (none)
    derivFcn: 'defaultderiv'
    divideFcn: (none)
    divideParam: (none)
    divideMode: 'sample'
    initFcn: 'initlay'
    performFcn: 'mse'
    performParam: .regularization, .normalization
    plotFcns:{}
    plotParams:{1x0 cell array of 0 params}
    trainFcn: (none)
    trainParam: (none)
weight and bias values:
    IW:{0x0 cell} containing 0 input weight matrices
    LW:{0x0 cell} containing 0 layer weight matrices
    b:{0x1 cell} containing 0 bias vectors
methods:
    adapt: Learn while in continuous use
    configure: Configure inputs & outputs
    gensim: Generate Simulink model
    init: Initialize weights & biases
    perform: Calculate performance
    sim: Evaluate network outputs given inputs
```

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate:　　outputs=net(inputs)

由上述运行结果可以得到，神经网络所有的属性都为0或空值，根据需要重新进行设置。主要需要重新设置以下5个方面。

·neural network object architecture（网络对象结构属性）。

·subobject structures（子对象结构属性）。

·functions（函数）。

·parameters（参数）。

·weight and bias values（权值和阈值）。

当设置或修改其中的任何一个属性值时，与之相关的属性都会自动改变，换句话说，并不是所有的属性都需要设置。

本节按照以下几个方面进行属性设置。

1.网络结构属性

·若网络有2 个输入向量，则numInputs=2。

·若网络有3 个网络层，则numLayers=3。

·若网络只有在第1、3网络层有阈值向量，第2网络层无阈值向量，则biasConnect=[1 0 1]。

·两个输入向量与网络层的连接：第1、2　网络层有来自输入向量p_1的连接；第2网络层还有来自p_2的连接；p_1，p_2与第3　网络层均无连接，故inputConnect=[1 0;1 1;0 0]。

·网络层之间的连接：第 3 个网络层有来自第 1、2 网络层以及自身的反馈连接，除此之外，无其他网络层的连接，故layerConnect=[0 0 0;0 0 0;1 1 1]。

·输出向量及其与网络层的连接：网络有2 个输出量，y^1来自第2 个网络层，y^2来自第3　个网络层，故outputConnect=[0 1 1]，而

numOutputs=2　　由输出向量与网络层的连接关系自动生成，不需要设置。

·输入和输出延迟量：在设置了输入层和网络层的连接权的延迟后，可以自动生成网络的输入和输出的延迟量，所以在此不必设置。

按照以上分析，以下列命令设置网络对象的结构。

```
net.numInputs=2;
net.numLayers=3;
net.biasConnect=[1 0 1]';
net.inputConnect=[1 0;1 1;0 0];
net.layerConnect=[0 0 0;0 0 0;1 1 1];
net.outputConnect=[0 1 1];
```

输入上述命令后，得到网络结构属性如下所示。

```
net =
  Neural Network
    name: 'Custom Neural Network'
    efficiency: .cacheDelayedInputs, .flattenTime,.memoryReduction
    userdata: (your custom info)
  dimensions:
    numInputs: 2
    numLayers: 3
    numOutputs: 2
    numInputDelays: 0
    numLayerDelays: 0
    numFeedbackDelays: 0
    numWeightElements: 0
    sampleTime: 1
  connections:
```

biasConnect: [1; 0; 1]

inputConnect: [1 0; 1 1; 0 0]

layerConnect: [0 0 0; 0 0 0; 1 1 1]

outputConnect: [0 1 1]

……

2.子对象结构属性

子对象结构在设置了网络结构属性后会自动生成。

subobjects:

inputs:{2x1 cell array of 2 inputs}

layers:{3x1 cell array of 3 layers}

outputs:{1x3 cell array of 2 outputs}

biases:{3x1 cell array of 2 biases}

inputWeights:{3x2 cell array of 3 weights}

layerWeights:{3x3 cell array of 3 weights}

……

但其属性值需根据自定义网络进行修改或重新设置。

（1）输入向量。

p_1有2个输入元素，其取值范围为0～2；p_2有5个输入元素，其取值范围为−2～2，只需要设置输入向量的取值范围。

net.inputs{1}.range=[0 2;0 2];

net.inputs{2}.range=[-2 2;-2 2;-2 2;-2 2;-2 2];

输入向量的 size 属性会自动设置。输入上面的设置后，再输入net.inputs{1}，net.inputs{2}，得到如下结果。

>> net.inputs{1}

ans =

Neural Network Input

name: 'Input'

```
    feedbackOutput: []
    processFcns:{}
    processParams:{1x0 cell array of 0 params}
    processSettings:{0x0 cell array of 0 settings}
    processedRange: [2x2 double]
    processedSize: 2
    range: [2x2 double]
    size: 2
    userdata: (your custom info)
>> net.inputs{2}
ans =
    Neural Network Input
    name: 'Input'
    feedbackOutput: []
    processFcns:{}
    processParams:{1x0 cell array of 0 params}
    processSettings:{0x0 cell array of 0 settings}
    processedRange: [5x2 double]
    processedSize: 5
    range: [5x2 double]
    size: 5
    userdata: (your custom info)
```

（2）网络层。

网络层在设置了网络结构和子对象结构属性后会自动生成，例如对第1个网络层而言有如下内容。

```
>> net.layers{1}
ans =
```

```
Neural Network Layer
name: 'Layer'
dimensions: 0
distanceFcn: (none)
distanceParam: (none)
distances: []
initFcn: 'initwb'
netInputFcn: 'netsum'
netInputParam: (none)
positions: []
range: []
size: 0
topologyFcn: (none)
transferFcn: 'purelin'
transferParam: (none)
userdata: (your custom info)
```

由上面代码可以看到，其属性值需要根据自定义网络进行修改或重新设置。第1层有4个神经元，第2层有3个神经元；而第3层的神经元由其输出向量决定，只有1个神经元。故只需要设置第 1、2 层的神经元数，net.layers{1}.size=4，net.layers{2}.size=3；选择initFcn=’initnw’；第 1 层的传输函数为 transferFcn=’tansig’ ，第 2 层的传输函数为 transferFcn=’logsig’。输入以下命令进行网络层设置。

```
net.layers{1}.size=4;
net.layers{1}.initFcn='initnw';
net.layers{1}.transferFcn='tansig';
net.layers{2}.size=3;
net.layers{2}.initFcn='initnw';
```

```
net.layers{2}.transferFcn='logsig';
net.layers{3}.size=1;
net.layers{3}.initFcn='initnw';
```

由此得到以下各个网络层的属性。

```
>> net.layers{1}
ans =
    Neural Network Layer
    name: 'Layer'
    dimensions: 4
    distanceFcn: (none)
    distanceParam: (none)
    distances: []
    initFcn: 'initnw'
    netInputFcn: 'netsum'
    netInputParam: (none)
    positions: []
    range: [4x2 double]
    size: 4
    topologyFcn: (none)
    transferFcn: 'tansig'
    transferParam: (none)
    userdata: (your custom info)
>> net.layers{2}
ans =
    Neural Network Layer
    name: 'Layer'
    dimensions: 3
```

```
    distanceFcn: (none)
    distanceParam: (none)
    distances: []
    initFcn: 'initnw'
    netInputFcn: 'netsum'
    netInputParam: (none)
    positions: []
    range: [3x2 double]
    size: 3
    topologyFcn: (none)
    transferFcn: 'logsig'
    transferParam: (none)
    userdata: (your custom info)
>> net.layers{3}
ans =
    Neural Network Layer
    name: 'Layer'
    dimensions: 0
    distanceFcn: (none)
    distanceParam: (none)
    distances: []
    initFcn: 'initnw'
    netInputFcn: 'netsum'
    netInputParam: (none)
    positions: []
    range: []
    size: 0
```

```
    topologyFcn: (none)
    transferFcn: 'purelin'
    transferParam: (none)
    userdata: (your custom info)
```

（3）输出向量。

在定义网络结构时自动生成输出向量属性如下。

```
>> net.outputs{2}
ans =
    Neural Network Output
    name: 'Output'
    feedbackInput: []
    feedbackDelay: 0
    feedbackMode: 'none'
    processFcns:{}
    processParams:{1x0 cell array of 0 params}
    processSettings:{0x0 cell array of 0 settings}
    processedRange: []
    processedSize: 0
    range: []
    size: 0
    userdata: (your custom info)
>>net.outputs{3}
ans =
    Neural Network Output
    name: 'Output'
    feedbackInput: []
    feedbackDelay: 0
```

feedbackMode: 'none'

processFcns:{}

processParams:{1x0 cell array of 0 params}

processSettings:{0x0 cell array of 0 settings}

processedRange: []

processedSize: 0

range: []

size: 0

userdata: (your custom info)

（4）目标向量。

在定义网络结构时自动生成目标向量属性如下。

```
>> net.targets{2}
ans =
    Neural Network Output
    name: 'Output'
    feedbackInput: []
    feedbackDelay: 0
    feedbackMode: 'none'
    processFcns:{}
    processParams:{1x0 cell array of 0 params}
    processSettings:{0x0 cell array of 0 settings}
    processedRange: []
    processedSize: 0
    range: []
    size: 0
    userdata: (your custom info)
>> net.targets{3}
```

Warning: SUBSREF used in an obsolete way.

> In obs_use at 18

In network.subsref at 98

"targets" is obsolete.

Use "outputs" to determine properties of outputs and targets.

ans =

Neural Network Output

name: 'Output'

feedbackInput: []

feedbackDelay: 0

feedbackMode: 'none'

processFcns:{}

processParams:{1x0 cell array of 0 params}

processSettings:{0x0 cell array of 0 settings}

processedRange: []

processedSize: 0

range: []

size: 0

userdata: (your custom info)

（5）阈值向量。

在定义网络结构时自动生成阈值向量属性如下。

>> net.biases{1}

ans =

Neural Network Bias

initFcn: (none)

learn: true

learnFcn: (none)

```
    learnParam: (none)
    size: 0
    userdata: (your custom info)
>> net.biases{2}
ans =
    []
>> net.biases{3}
ans =
    Neural Network Bias
    initFcn: (none)
    learn: true
    learnFcn: (none)
    learnParam: (none)
    size: 0
    userdata: (your custom info)
```

（6）输入权值向量。

在定义网络结构时自动生成输入权值向量属性如下。

```
>> net.inputWeights{1,1}
ans =
    Neural Network Weight
    delays: 0
    initFcn: (none)
    initSettings: (none)
    learn: true
    learnFcn: (none)
    learnParam: (none)
    size: [0 2]
```

```
    weightFcn: 'dotprod'
    weightParam: (none)
    userdata: (your custom info)
>> net.inputWeights{1,2}
ans =
    []
>> net.inputWeights{2,1}
ans =
    Neural Network Weight
    delays: 0
    initFcn: (none)
    initSettings: (none)
    learn: true
    learnFcn: (none)
    learnParam: (none)
    size: [0 2]
    weightFcn: 'dotprod'
    weightParam: (none)
    userdata: (your custom info)
>> net.inputWeights{2,2}
ans =
    Neural Network Weight
    delays: 0
    initFcn: (none)
    initSettings: (none)
    learn: true
    learnFcn: (none)
```

```
    learnParam: (none)
    size: [0 5]
    weightFcn: 'dotprod'
    weightParam: (none)
    userdata: (your custom info)
>> net.inputWeights{3,1}
ans =
    []
>> net.inputWeights{3,2}
ans =
    []
```

根据图13-1所示的神经网络，第1层和第2层与输入向量的连接权有延迟，第3层与自身输出的连接权有延迟，输入以下命令重新设置。

```
net.inputWeights{2,1}.delays=[0 1];
net.inputWeights{2,2}.delays=1;
```

（7）网络层权值向量。

在定义网络结构时自动生成网络层权值向量属性。

```
>> net.layerWeights{1,1}
ans =
    []
>> net.layerWeights{1,2}
ans =
    []
>> net.layerWeights{1,3}
ans =
    []
>> net.layerWeights{2,1}
```

```
ans =
    []
>> net.layerWeights{2,2}
ans =
    []
>> net.layerWeights{2,3}
ans =
    []
>>
>> net.layerWeights{3,1}
ans =
    Neural Network Weight
    delays: 0
    initFcn: (none)
    initSettings: (none)
    learn: true
    learnFcn: (none)
    learnParam: (none)
    size: [0 0]
    weightFcn: 'dotprod'
    weightParam: (none)
    userdata: (your custom info)
>> net.layerWeights{3,2}
ans =
    Neural Network Weight
    delays: 0
    initFcn: (none)
```

initSettings: (none)

learn: true

learnFcn: (none)

learnParam: (none)

size: [0 0]

weightFcn: 'dotprod'

weightParam: (none)

userdata: (your custom info)

>> net.layerWeights{3,3}

ans =

Neural Network Weight

delays: 0

initFcn: (none)

initSettings: (none)

learn: true

learnFcn: (none)

learnParam: (none)

size: [0 0]

weightFcn: 'dotprod'

weightParam: (none)

userdata: (your custom info)

根据图13-1所示的神经网络，第3层与自身输出的连接权有延迟，输入以下命令重新设置。

net.layerWeights{3,3}.delays=1;

3.函数属性

通过以下命令设置函数属性。

net.initFcn=’initlay’;

net.performFcn=’mse’;

net.trainFcn=’trainlm’;

输入net，观察函数属性结果。

…

functions:

adaptFcn: (none)

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: (none)

divideParam: (none)

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mse'

…

4.参数属性

当确定了函数属性后，函数的参数属性以各函数的默认值自动生成。输入net，观察函数属性结果。

…

performParam: .regularization, .normalization

plotFcns:{}

plotParams:{1x0 cell array of 0 params}

trainFcn: 'trainlm'

trainParam: .showWindow, .showCommandLine, .show, .epochs,.time, .goal, .min_grad, .max_fail, .mu, .mu_dec, .mu_inc, .mu_max

…

5.权值和阈值属性

当网络结构属性和子对象属性确定下来以后，网络权值和阈值的结构就确定了。

```
>> net.IW
ans =
    [0x2 double]          []
    [0x4 double]    [0x5 double]
    []          []
>> net.LW
ans =
    []    []    []
    []    []    []
    []    []    []
>> net.b
ans =
    [0x1 double]
    []
    [0x1 double]
```

由于网络还未进行初始化，因此所有权值和阈值的具体数据全为0。

现在完成对图13-1所示网络的创建工作，键入net，可以查看所定义的网络相关属性。

```
>> net
net =
    Neural Network
        name: 'Custom Neural Network'
        efficiency: .cacheDelayedInputs, .flattenTime, .memoryReduction
        userdata: (your custom info)
```

dimensions:

numInputs: 2

numLayers: 3

numOutputs: 2

numInputDelays: 1

numLayerDelays: 0

numFeedbackDelays: 0

numWeightElements: 0

sampleTime: 1

connections:

biasConnect: [1; 0; 1]

inputConnect: [1 0; 1 1; 0 0]

layerConnect: [0 0 0; 0 0 0; 1 1 1]

outputConnect: [0 1 1]

subobjects:

inputs:{2x1 cell array of 2 inputs}

layers:{3x1 cell array of 3 layers}

outputs:{1x3 cell array of 2 outputs}

biases:{3x1 cell array of 2 biases}

inputWeights:{3x2 cell array of 3 weights}

layerWeights:{3x3 cell array of 3 weights}

functions:

adaptFcn: (none)

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: (none)

divideParam: (none)

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mse'

performParam: .regularization, .normalization

plotFcns:{}

plotParams:{1x0 cell array of 0 params}

trainFcn: 'trainlm'

trainParam: .showWindow, .showCommandLine, .show, .epochs, .time, .goal, .min_grad, .max_fail, .mu, .mu_dec, .mu_inc, .mu_max

weight and bias values:

IW:{3x2 cell} containing 3 input weight matrices

LW:{3x3 cell} containing 3 layer weight matrices

b:{3x1 cell} containing 2 bias vectors

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate: [outputs,inputStates]= net(inputs,inputStates)

13.1.2 网络的初始化、训练和仿真

1.自定义神经网络的初始化

自定义神经网络可以调用 init 函数，以定义的权值和阈值初始化函数对网络的权值和阈值进行初始化。

net=init(net)

初始化后的权值和阈值已经不再全部为0，其结果如下。

```
>> net.IW
ans =
    [4x2 double]              []
    [3x4 double]    [3x5 double]
    []              []
>> net.IW{1,1}
ans =
   -1.8602    1.9370
   -0.2733    2.7897
   -1.6207    2.2832
   -2.4792   -1.3016
>> net.IW{2,1}
ans =
    0.7938   -0.9785   -0.6925    0.8872
   -0.1758    0.0997   -0.4962   -0.5885
    0.6720    0.7943   -0.2092    0.2925
>> net.IW{2,2}
ans =
   -0.7030    0.2559    0.1398    0.2044   -0.0133
   -0.1059   -0.4524    0.2517    0.1141    0.1900
    0.5950   -0.5657   -0.8580-0.2808   -0.0336
>> net.LW
```

```
ans =
    []                    []                []
    []                    []                []
    [1x4 double]    [1x3 double]    [0.1923]
>> net.LW{3,1}
ans =
   -0.0930   -0.1998   -0.1268    1.0950
>> net.LW{3,2}
ans =
    0.5869   -0.8824    0.1282
>> net.LW{3,3}
ans =
    0.1923
>> net.b
ans =
    [4x1 double]
    []
    [    0.0114]
>> net.b{1}
ans =
    3.0908
   -1.5492
   -1.5989
    1.0725
>> net.b{2}
ans =
    []
```

```
>> net.b{3}
ans =
   0.0114
```

2.自定义神经网络的训练

自定义网络可以调用train函数，以定义的网络训练函数对网络进行训练，由于第3层定义了目标向量，所以训练前必须定义输入向量和目标向量。

```
p={[0;0] [2;0.5];[2;-2;1;0;1] [-1;-1;1;0;1]};
t={1 -1};
net=train(net,p,t)
```

得到自定义神经网络训练框如图13-2所示。

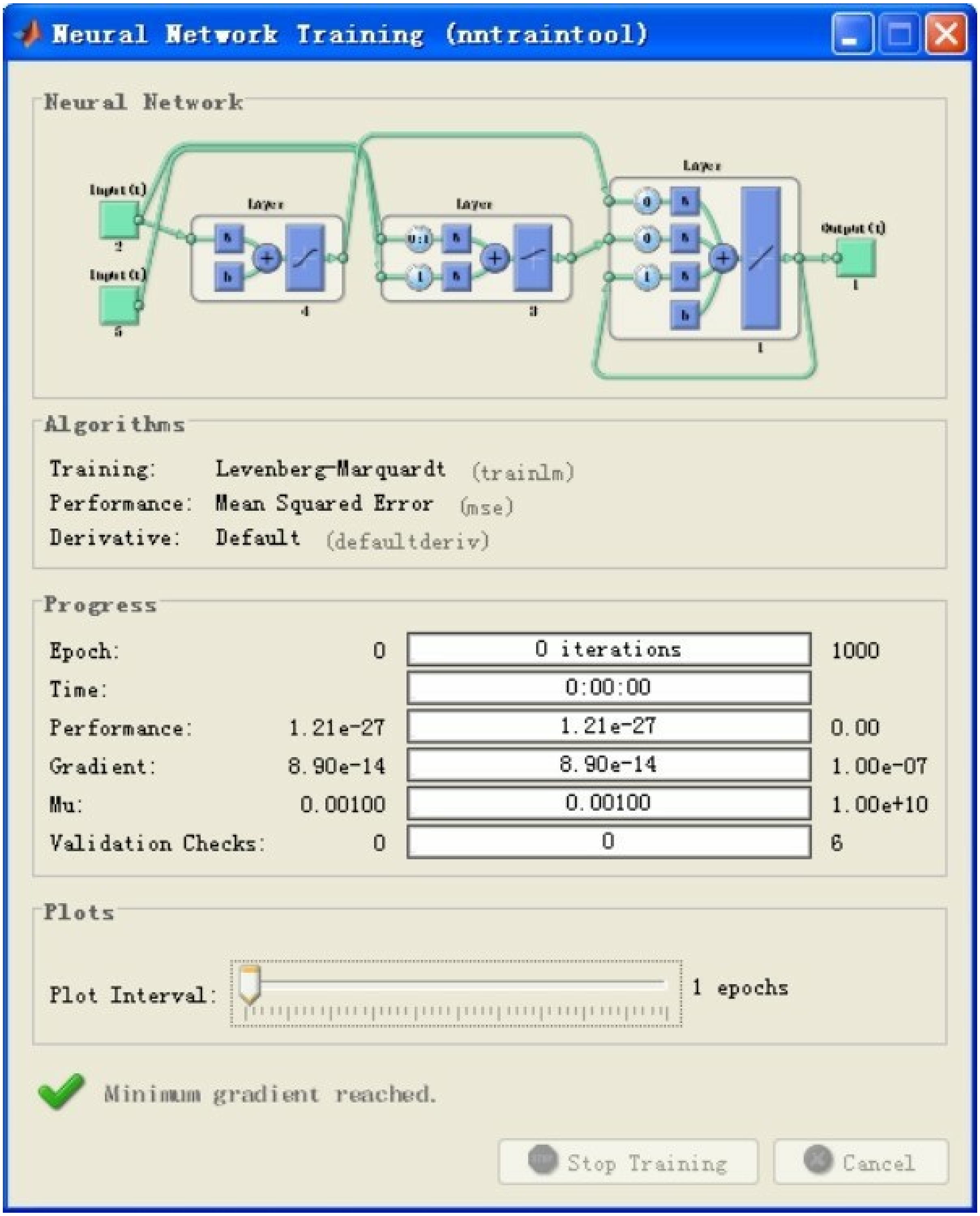


图13-2 自定义神经网络训练框图

训练后得到的网络如下所示。

```
>>net =
```

Neural Network

name: 'Custom Neural Network'

efficiency: .cacheDelayedInputs, .flattenTime,

.memoryReduction, .flattenedTime

userdata: (your custom info)

dimensions:

numInputs: 2

numLayers: 3

numOutputs: 1

numInputDelays: 1

numLayerDelays: 1

numFeedbackDelays: 1

numWeightElements: 48

sampleTime: 1

connections:

biasConnect: [1; 0; 1]

inputConnect: [1 0; 1 1; 0 0]

layerConnect: [0 0 0; 0 0 0; 1 1 1]

outputConnect: [0 0 1]

subobjects:

inputs:{2x1 cell array of 2 inputs}

layers:{3x1 cell array of 3 layers}

outputs:{1x3 cell array of 1 output}

biases:{3x1 cell array of 2 biases}

inputWeights:{3x2 cell array of 3 weights}

layerWeights:{3x3 cell array of 3 weights}

functions:

adaptFcn: (none)

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: (none)

divideParam: (none)

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mse'

performParam: .regularization, .normalization

plotFcns:{}

plotParams:{1x0 cell array of 0 params}

trainFcn: 'trainlm'

trainParam: .showWindow, .showCommandLine, .show, .epochs, .time, .goal, .min_grad, .max_fail, .mu, .mu_dec, .mu_inc, .mu_max

weight and bias values:

IW:{3x2 cell} containing 3 input weight matrices

LW:{3x3 cell} containing 3 layer weight matrices

b:{3x1 cell} containing 2 bias vectors

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate: [outputs,inputStates,layerStates]= net(inputs,inputState,layerStates)

13.2 自定义函数

神经网络工具箱允许创建并运用很多种类的自定义函数，用户可以根据需要，在初始化、仿真及训练中运用多种方法，实现对网络的自行调整。

这些函数主要用来进行神经网络的初始化、学习、训练和仿真，主要分为以下三类。

（1）初始化函数。

·网络初始化函数。

·层初始化函数。

·权值和阈值初始化函数。

（2）学习函数。

·网络训练函数。

·网络自适应函数。

·网络性能函数。

·权值和阈值学习函数。

（3）仿真函数。

·传递函数。

·网络输入函数。

·权值函数。

自定义函数需要对所定义的函数本身、函数变量的数据格式、函数的限制条件和编程技巧等很熟悉，本书就自定义函数的创建及方法不做详细介绍，有兴趣的读者可以查阅MATLAB神经网络工具箱的帮助文件。

13.2.1 初始化函数

初始化函数包括网络、层、权值和阈值 3 种初始化函数。下面分别介绍如何定义这 3种函数。

1.网络初始化函数

网络初始化函数将所有的权值和阈值设置为一个适当的值，作为网络训练或者自适应条件的初始点。一旦定义了网络初始化函数，就可以嵌入到某一网络中。

假设定义网络的初始化函数为csh()，并嵌入到某一网络中，则可以用下面的语句实现。

net.initFcn=’csh’

这样，在调用init()初始化网络时，都可以应用此时设定的网络初始化函数进行初始化。

net=init(net)

网络初始化函数编制完成后，接受某一网络，并且在初始化处理以后，再返回一个网络。

net=csh(net,i)

自定义网络初始化函数可以根据要求对权值和阈值进行任意设置。

2.层初始化函数

层初始化函数将某层所有权值和阈值设置为一个适当的值，作为网络训练或者自适应调节的初始点。一旦定义了层初始化函数，就可以嵌入到网络任意一层上。假设定义的层初始化函数为csh()，并嵌入网络第3层上，则应用下面语句实现。

net.layers{3}.initFcn=’csh’

如果网络初始化函数net.initFcn设置为工具箱函数initlay()，自定义层初始化函数用来对层进行初始化。那么在调用init()初始化网络时，都可以应用此时设定的层初始化函数进行初始化。

net=init(net)

层初始化函数编制完成后，接受网络和层的标号作为输入i，并且在对第i层初始化处理以后，再返回网络。

net=csh(net,i)

自定义层初始化函数可以根据要求对层的权值和阈值进行任意设置。

3.权值和阈值初始化函数

权值和阈值初始化函数将所有的权值和阈值设置为一个适当的值，作为网络训练或者自适应调节的初始点。一旦定义了权值和阈值初始化函数，就可以嵌入到网络任意权值和阈值上。

假设定义了权值和阈值初始化函数为cshf()，并嵌入网络第2层的阈值、第1层输入到第2层的权值上，则应用下面的语句实现。

net=init(net)

权值和阈值初始化函数编制完成后，可以应用如下方式进行调用。

W=rands(S,PR)

b=rands(S)

其中，S为层神经元数目；PR为R个输入向量的最大/最小值矩阵。

13.2.2 学习函数

与网络学习、权值和阈值调整有关的学习函数有4种，分别是训练函数、自适应函数、性能函数、权值和阈值学习函数。下面分别介绍如何自定义生成这些函数。

1.训练函数

训练函数是常用的学习函数。学习函数循环地将输入向量应用于网络中，每次都能更新网络，直到达到训练目标为止。训练停止的条件可以是最大学习次数、最小的误差梯度或者训练精度等。

一旦定义了训练函数，就可以嵌入到某一网络上。假设定义了训练函数为ts()，并嵌入某个网络中，则应用下面的语句实现。

net.trainFcn=’ts’

这样，训练网络时，都可以应用此时设定的训练函数。

[net,tr]=train(NET,P,T,Pi,Ai)

训练函数编制完成后，可以应用如下方式进行调用。

[net,tr]=ts(net,Pd,T1,Ai,Q,TS,VV,TV)

自定义训练函数需要提供如下信息。

·version：神经网络工具箱版本。

·pdefaults：默认参数。

自定义训练函数可以根据所设定的任意方式更新网络的权值和阈值。

2.自适应函数

自适应函数在每个输入时间段内都要更新网络，并进行仿真。一旦定义了自适应函数，就可以嵌入到某一网络上。假定自适应函数为zsy()，并嵌入某个网络中，则应用下面语句实现。

net.adaptFcn=’zsy’;

这样，自适应调节网络时，都可以应用此时设定的自适应函数。

[net,Y,E,Pf,Af]=adapt(NET,P,T,Pi,Ai)

自适应函数编制完成后，可以应用如下方式进行调用。

[net,Ac,El]=zsy(net,Pd,Ti,Ai,Q,TS)

自定义自适应函数需要提供如下信息。

·version：神经网络工具箱的版本。

·Pdefaults：默认参数。

自定义自适应函数可以根据所设定的任意方式更新网络的权值和阈值。

3.性能函数

性能函数是学习训练时，期望达到最优的指标，通过改变权值和阈值使性能函数最优，改善网络性能。一旦定义了性能函数，就可以嵌入到某一网络上。假定性能函数为 xn()，并嵌入某个网络中，则应用下面语句实现。

net.performFcn=’xn’

这样训练网络时，都可以应用此时设定的性能函数进行优化。

[net,tr]=train(NET,P,T,Pi,Ai)

[net,Y,E,Pf,Af]=adapt(NET,P,T,Pi,Ai)

性能函数编制完成后，可以应用如下方式进行调用。

perf=xn(E,X,PP)

其中，E为期望值矩阵，X为网络权值和阈值，PP为网络参数。

如果E是细胞数组，则需要首先转换成矩阵形式，然后再进行调用，X和PP的值通过网络得到。

E=cell2mat(E)；

perf=xn(E,net);

X=getx(net);

PP=net.performParam;

自定义性能函数需要提供如下信息。

·version：神经网络工具箱的版本。

·deriv：相关导数函数名。

·Pdefaults：默认参数。

4.权值和阈值学习函数

权值和阈值学习函数与某些训练或者自适应调节函数一起使用，用于学习训练中更新权值和阈值。一旦定义了权值和阈值学习函数，就可以嵌入到网络任意一层上。

13.2.3 仿真函数

自定义仿真函数包括传递函数、网络输入函数、权值函数，下面分别介绍这3种仿真函数及其相应的导数函数。

1.传递函数

传递函数根据给定的网络输入向量 N，计算某层的输出向量 A，网络输入向量和输出向量必须具有相同的维数。一旦定义了传递函数，就可以嵌入到网络中任意一层上。假定传递函数为cd()，并嵌入网络第4层上，则应用下面语句实现。

net.layers{4}.transferFcn=’cd’

这样，在对网络进行仿真时，都可以应用此时设定的传递函数。

[Y,Pf,Af]=sim(net,P,Pi,Ai)

传递函数编制完成后，可以应用如下方式进行调用。

A=cd(X)

其中，X为网络输入向量，A为函数返回值，即网络输出向量。

2.网络输入函数

网络输入函数根据给定的加权输入向量 Z，计算某层的网络输入向量 N，网络输入向量和加权输入向量必须具有相同的维数。一旦定义了网络输入函数，就可以嵌入到网络中的任意一层上。

假定定义网络输入函数为input()，并嵌入到第4层上，则应用下面语句。

net.layers{4}.netInputFcn=’input’

这样，在对网络进行仿真时，都可以应用此时设定的网络输入函数。

[Y,Pf,Af]=sin(net,P,Pi,Ai)

网络输入函数编制完成后，可以应用如下方式进行调用。

N=input(X1,X2,…)

其中，X1,X2…为加权输入向量，N为函数返回值，即网络输入向量。

3.权值函数

权值函数根据给定的输入向量P及权值矩阵W，计算一个加权的输入向量Z，一旦定义了权值函数，就可以嵌入到网络中任意输入权值和层权值上。

假定定义权值函数为wgh()，并嵌入网络第1　个输入到第4层的权值上，则应用以下语句。

```
net.inputWeights{1,4}.weightFcn='wgh'
```

13.3 本章小结

本章重点介绍了自定义神经网络的创建、初始化、训练和仿真，并举例说明。对于自定义函数，本章只做了简单的内容介绍，详细说明可以参考MATLAB神经网络工具箱的帮助文件。

第14章 随机神经网络

随机神经网络是一类功能强大的人工神经网络，在优化计算和模拟学习领域中发挥着重要的作用。人们通常所说的人工神经网络一般有两种。一种是采用随机性神经元激活函数的随机神经网络；另一种是采用随机型加权连接，即在普通人工神经网络中加入适当随机噪声的Boltzmann（波耳兹曼）机。

本章主要介绍随机神经网络的基本思想，模拟退火算法和Boltzmann 机以及随机神经网络的实际应用。

学习目标：

·理解随机神经网络基本思想

·熟悉模拟退火算法

·熟悉Boltzmann 机

·掌握随机神经网络的应用

14.1 随机神经网络的基本思想

人工神经网络分为很多种类，随机神经网络与其他两种神经网络特性比较见表14-1。

表14-1 随机神经网络与其他网络比较

名称	网络类型	网络结构	学习算法
BP 神经网络	多层前向网络	含输入层、隐层、输出层。层内神经元无连接	网络按误差减少的最大梯度方向调整权值
Hopfield 神经网络	反馈神经网络	单层神经网络，层内神经元全互联	网络按照其用途来设计或训练网络权值
Boltzmann 机	随机神经网络	含输入部、输出部和中间部。神经元互联	网络向误差减小的方向运行概率大，但也可能向误差增大方向运行

随机神经网络是统计力学思想引入神经网络研究的结果，其基本思想是，网络向误差或能量函数减小方向运行的概率大，同时向误差或能量函数增大方向运行的概率存在，这样网络跳出局部极小点的可能性存在，而且向全局最小点收敛的概率最大。

BP神经网络是多层前向网络，其基本思想是通过对网络参数（连接权、输出阈值）的逐步调整实现网络学习。它是在提供给网络的学习模式集合的全局误差按梯度下降的条件下达到网络记忆目的。当学习过程进行到全局误差开始有上升的趋势时就停止了，这往往导致网络误差陷入局部极小值，而达不到全局最小点。

BP神经网络这种算法被形象地称为“贪心”算法，即急于找到最小解，结果则是欲速则不达。

对于Hopfield网络，其工作规则也是使网络的能量函数朝梯度下降的方向变化，即随着网络状态的不断更新，网络能量函数单调下降，其结果也往往是使网络陷入局部极小值，最终得不到网络的最优解。

在用BP网络和Hopfield网络进行最优化的计算时，由于限定条件的

不足，往往会使网络稳定在误差或能量函数的局部最小点，而不是全局最小点，即所得的解不是最优解。

14.2 模拟退火算法

模拟退火算法（Simulated Annealing Algorithm）来源于固体退火原理，就是模拟金属构件退火过程的一种算法，由Metropolis 算法和退火过程（Annealing Procedure，AP）组成。

金属或某类固体物质退火处理过程如下。

·先用高温将其加热熔化，使其中的粒子可以自由运动。

·逐渐降低温度，粒子的自由运动趋势也逐渐减弱，并逐渐形成低能态晶格。

·若在凝结点附近温度下降的速度足够慢，则金属或固体物质一定会形成最低能量的基态，即最稳定结构状态。

实际上，在整个降温的过程中，各个粒子都可能经历了由高能态向低能态、有时又暂时由低能态向高能态最终趋向低能态的变化过程。

如果把神经网络的状态看作金属内部的“粒子”，把网络在各个状态下的能量函数 E看作是粒子所处的能态；在算法中设置一种控制参数T，当T较大时，网络能量由低向高变化的可能性也较大；随着T的减小，这种可能性也减小。如果把这个参数看作温度，让其由高慢慢地下降，则整个网络状态变化过程就完全模拟了金属的退火过程，当参数了下降到一定程度时，网络将收敛于能量的最小值。

模拟退火算法中，如下两点是算法的关键。

·控制参数T。

·能量由低向高变化的可能性。

这两点必须结合起来考虑，当T大时，可能性也大，T小时，可能性也小，把“可能性”当作参数T的函数。

模拟退火的最初目的是寻找代表复杂系统的代价函数的全局最小

值。因此，这种方法为解决凹平面最优问题提供了一个强有力的工具，其中心思想在于，在用模拟退火最优化一个复杂系统（如一个拥有很多自由度的系统）时，误差或能量函数绝大部分的时间在下降，但不是一直下降，即误差或能量函数的总趋势向减小的方向变化，但有时也向增大的方向变化，这样可跳出局部极小点，向全局最小点收敛。

模拟退火与传统迭代最优算法的比较如下。

·当系统在非零温度下时，从局部最优中跳出是非常可能的，因此不会陷入局部最优。

·系统最终状态的总特征可以在较高温度下看到，而状态的好的细节却在低温下表现，因此，模拟退火是自适应的。

14.2.1 模拟退火算法的原理

1.Metropolis抽样过程

假定一随机变量在某一时刻的状态为v_i。在另一时刻的状态为v_j。假设这种状态的转移满足以下条件。

$$P(v_j \mid v_i) = P(v_i \mid v_j)$$

ΔE表示系统从状态v_i转移至状态v_j所引起的能量差。

如果能量差ΔE为负，这种转移就导致状态能量的降低，这种转移就被接受。然后新状态将作为算法下一步的起始点。

若能量差为正，算法在这一点进行概率操作。首先，选定一个在[0,1]内服从均匀分布的随机数ξ。如果$\xi < e^{-\Delta E/T}$，则接受这种转移。否则，拒绝这种转移；即在算法的下一步中拒绝旧的状态。如此反复，达到系统在此温度下的热平衡。

这个过程称作Metropolis抽样过程。Metropolis抽样过程就是在一确定温度下，使系统达到热平衡的过程。

2.退火过程（降温过程）

在Metropolis抽样过程中温度T缓慢地降低。模拟退火过程就是通过T参数的变化使状态收敛于最小能量处。因而，T　参数的选择对于算法最后的结果有很大影响。初始温度和终止温度设置得过低或过高都会延长搜索时间。

降温步骤太快，往往会漏掉全局最优点，使算法收敛至局部最优点。降温步骤太慢，则会大大延长搜索全局最优点的计算时间，从而难以实际应用。因此，T 可以理解为一个控制参数。

为寻找在有限时间逼近全局最优的模拟退火算法，设置了许多控制算法收敛的参数。在退火过程中指定了有限的退火温度值和在每一温度下的转移数目。Kirlpatrick等人在退火步骤中设定的参数如下。

（1）初始温度值。初始温度值T_0要选得足够高，保证模拟退火算法中所有可能的转移都能被接受。

（2）温度的下降。原先使用指数函数实现温度的下降。但是这种方法使降温幅度过小，从而延长搜索时间。在实际中，通常使用下式。

$$T_k = \lambda T_{k-1}, k = 1, 2, \cdots$$

此处λ是一小于却接近于1的常数。λ通常的取值在0.8～0.99之间。在每一温度下，实验足够多的转移次数。

（3）终止温度。如果在连续的若干个温度下没有可接受的新状态，系统冻结或退火停止。

模拟退火尤其适合解决组合优化问题，下面以模拟退火算法解决组合优化问题来进一步介绍模拟退火算法的步骤。

14.2.2 模拟退火算法用于组合优化问题

许多工程上和理论上的问题，其目标都是在一个很大的解空间中寻求一个最优解，这些问题统称为组合优化问题。

在许多组合优化问题中，一个解通常是满足一定规则的一些离散对象的排列，所有这些解的集合叫做解空间。通常用一个“代价函数”C(x)来衡量一个解的优劣，目标就是选择一个解使其代价函数C(x)最小，如TSP问题、大规模集成电路布局布线问题等。

模拟退火与组合优化问题的对应关系见表14-2。

表14-2 模拟退火与组合优化问题的对应关系

模拟退火算法	组合优化问题
样本	问题举例
状态	解
能量	代价函数
温度	控制参数
热平衡时的能量	最小代价
热平衡状态	最优解

设$V=\{V_1,V_2,\ldots,V_n\}$为所有可能的组合（或状态）所构成的集合。C(·)是V的函数，且$C(V_i)$.0，反映取状态V_i为解的代价，目标是寻找 $V^*\in V$ 使

$$C(V^*)=\min C(V_i) \qquad V_i\in V$$

模拟退火算法应用于组合优化问题的基本思想就是把每种组合状态V_i看成某一物质体系的微观状态，$C(V_i)$可看成该物质体系在V_i下的能量，温度T为控制参数。

让T从一个足够高的值慢慢下降，对于每个T，对当前状态V作随机扰动产生一个新状态V′，计算其增量$\Delta C=C(V)-C(V')$，并以概率$e^{-\Delta C/kT}$接受V′作为新的当前状态。根据统计力学的知识，当重复如此随机扰动足够次数后，状态V_i的出现概率如下。

$$P = e^{-C(V_i)/kT} \Big/ \sum_i e^{-C(V_i)/kT} \qquad k\text{为Botzmann常数}$$

服从Boltzmann分布。

模拟退火用于组合优化问题算法实现步骤如下。

第一步、初始化。依据所要解决的组合优化问题，确定代价函数C(i)的表达式，随机选择初始状态V=V(0)，设定初始温度T_0，终止温度T_{final}，概率阈值ξ。

第二步、Metropolis抽样过程。

① 在温度 T 下依据某一规定的方式，根据当前解所处的状态 V，产生一个近邻子集N(V)（可包括V，也可不包括V），在N(V)内随机寻找一个新状态S′作为下一个当前解的候选解，计算

$$\Delta C = C(V) - C(V')$$

。

② 若ΔC'<0，则V=V'，作为下一状态；若ΔC′>0，则计算概率$e^{-\Delta C'/T}$，若其大于给定概率阈值ξ，则取下一状态为V=V'，否则，保留这一状态。

③ 按某一给定的收敛算法检查算法在温度T下是否应停止，若符合收敛条件则表示已达到热平衡，转向第三步的退火过程，若不符合收敛条件，则转向①继续迭代，直至在此温度下收敛。

第三步、退火过程。

按照一定的降温方法得到一个新的温度T，检查T是否小于给定的温度终止阈值T_{final}。若小于，则退火过程结束，当前状态V即为算法最终输出解。若温度T大于等于给定阈值，则转至Metropolis抽样过程，在新的温度下搜索状态。

注意：在上述退火过程中，模拟退火算法是否能达到能量 E 的最小值，取决于 T_0是否足够高，和T下降得是否充分慢，以及对每个T时系

统是否稳定。

14.2.3 退火算法的参数控制

（1）T_0的选择方法。

·均匀随机抽样{V_i}，取此时$C(V_i)$的方差为T_0。

·在所有可能的组合状态中，选两个状态使ΔC'最大，取T_0为ΔC'的若干倍。

·按经验给出。

（2）退火过程中T_{final}的选取方法。

·依据经验确定。

·检验系统的熵是已否达到最小，若达到最小，即可认为温度已达到终止温度。

·T下降n次后都没有改善，即可认为能量已降到最低，没有必要再降温。

（3）Metropolis抽样过程的收敛算法。

·检验目标函数C(·)的均值是否稳定。

·继续若干步，C(·)变化很小（设定阈值）。

·按一个固定步数抽样。

（4）降温方法的确定。

根据Kirlpatrick的方法令$\lambda T \to T, \quad \lambda \in [0.8, 0.99]$。

模拟退火算法是一种通用的随机搜索算法，它可用于解决众多的优化问题，并已经广泛应用于其他领域。如VLSL设计、图像识别等。

当待解决的问题复杂性较高，而且规模较大时，在对问题的领域知识甚少的情况下，采用模拟退火算法最合适。因为模拟退火算法不像其他确定型启发式算法那样，需要依赖于问题的领域知识来提高算法的性

能。

但是，从另一方面来说，已知有关待解决问题的一些知识后，模拟退火算法却无法充分利用它们，这使得模拟退火算法的优点就成了缺点。

模拟退火算法具有跳出局部最优陷阱的能力，因此被Ackley、Hinton和Sejnowski用作Boltzmann机学习算法，从而使Boltzmann机克服了Hopfield网络经常收敛到局部最优点的缺点。在Boltmann机中，即使系统落入局部最优的陷阱，经过一段时间后，它还能重新跳出来，使系统最终往全局最优点的方向收敛。

模拟退火算法在求解规模较大的实际问题时，往往存在以下缺点。

① 收敛速度比较慢。

② 尽管理论上只要计算时间足够长，模拟退火法就可以保证以概率1 收敛于全局最优点。但是在实际算法的实现过程中，由于计算速度和时间的限制，在优化效果和计算时间二者之间存在矛盾，因而难以保证计算结果为全局最优点，优化效果不甚理想。

③ 在每一温度下很难判定是否达到了平衡状态。

为此，人们对模拟退火算法提出了各种各样的改进，其中包括并行模拟退火算法、快速模拟退火算法（Cauchy机）和对模拟退火算法中各个函数和参数的重新设计等。

14.3 Boltzmann机

20世纪80年代，Ackley和Sejnowski等人以模拟退火思想为基础，对Hopfield网络引入了随机机制，推出Boltzmann机。

Boltzmann 机是第一个受统计力学启发的多层学习机，它是典型的随机神经网络。其命名来源于 Boltzmann 机在统计力学中的早期工作和网络本身的动态分布行为（其平衡状态服从Boltzmann分布），其运行机制服从模拟退火算法。

14.3.1 Boltzmann机的网络结构

Boltzmann 机由输入部、输出部和中间部构成。输入神经元和输出神经元可称为显见神经元，它们是网络与外部环境进行信息交换的媒介。中间部的神经元称为隐见神经元，它们通过显见神经元与外部进行信息交换。

Boltzmann机的网络结构如图14-1所示。

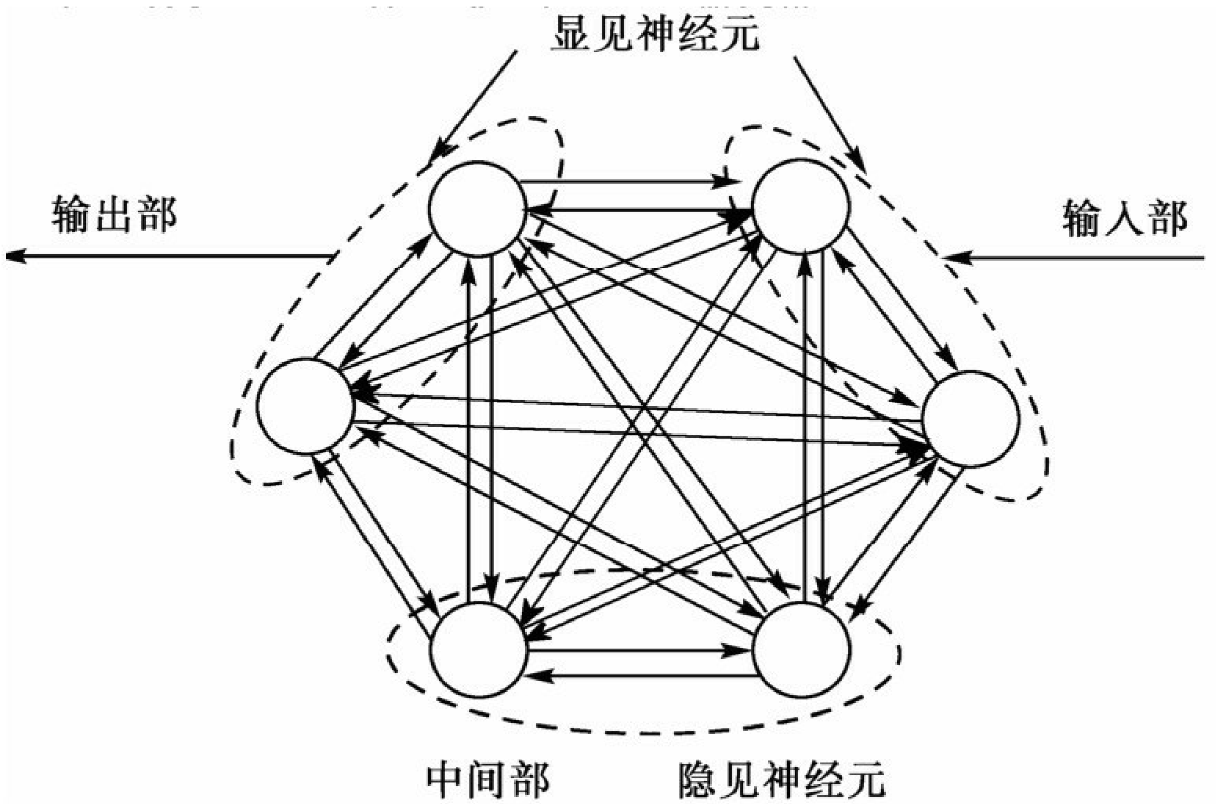


图14-1 Boltzmann机的网络结构

每一对神经元之间的信息传递是双向对称的，即$w_{ij}= w_{ji}$，而且自身无反馈即$w_{ii}=0$。学习期间，显见神经元将被外部环境“约束”在某一特定的状态，而中间部隐见神经元则不受外部环境约束。

Boltzmann机中单个神经元如图14-2所示，其中v为输入，w为权值，b为阈值，y为输出。

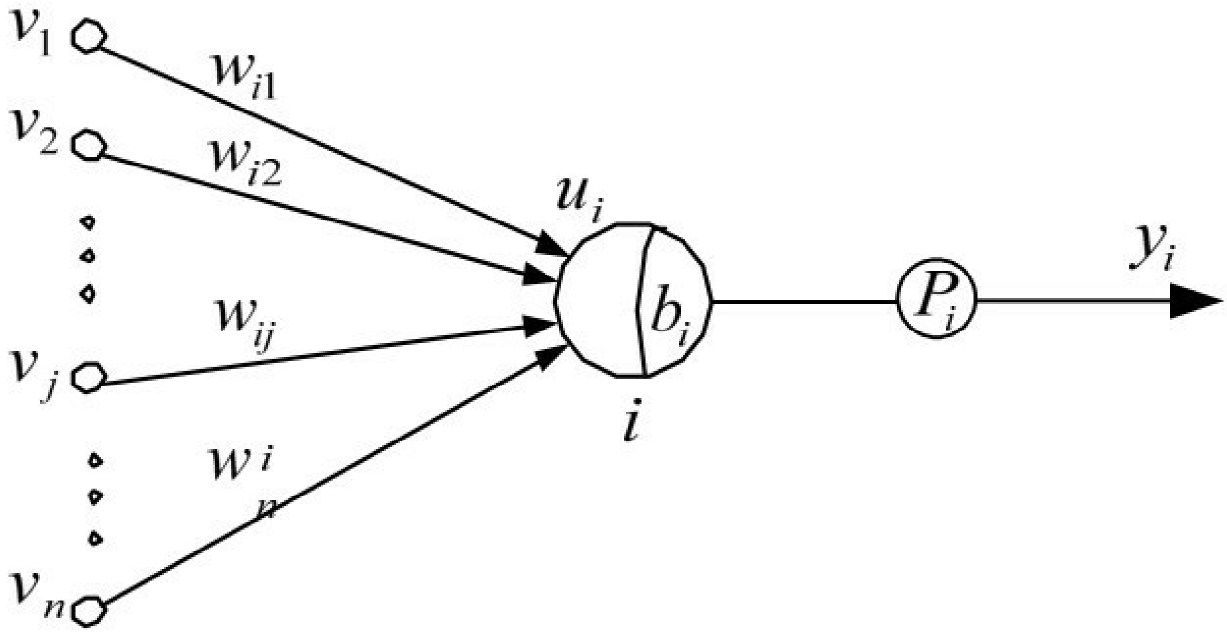


图14-2 Boltzmann机单个神经元

Boltzmann机中每个神经元的兴奋或抑制具有随机性，其概率取决于神经元的输入。

神经元i的全部输入信号的总和为u_i。

$$u_i = \sum_{j}^{n} w_{ij} v_j + b_i$$

式中b_i是该神经元的阈值。

可以将b_i归并到总的加权和中去。

$$u_i = \sum_{j}^{n} w_{ij} v_j$$

神经元的输出v_i依概率取1或0。

v_i取1的概率：$P(v_i = 1) = 1 / (1 + e^{-u_i / T})$

v_i取0的概率：

$$P(v_i = 0) = 1 - P(v_i = 1) = e^{-u_i/T} \cdot P(v_i = 1)$$

由此可见，v_i取1的概率受如下两个因素的影响。

·u_i越大v_i则取1 的概率越大，而取0 的概率越小。

·参数T 称为“温度”，在不同的温度下v_i取1的概率P 随u_i的变化如图14-3 所示。

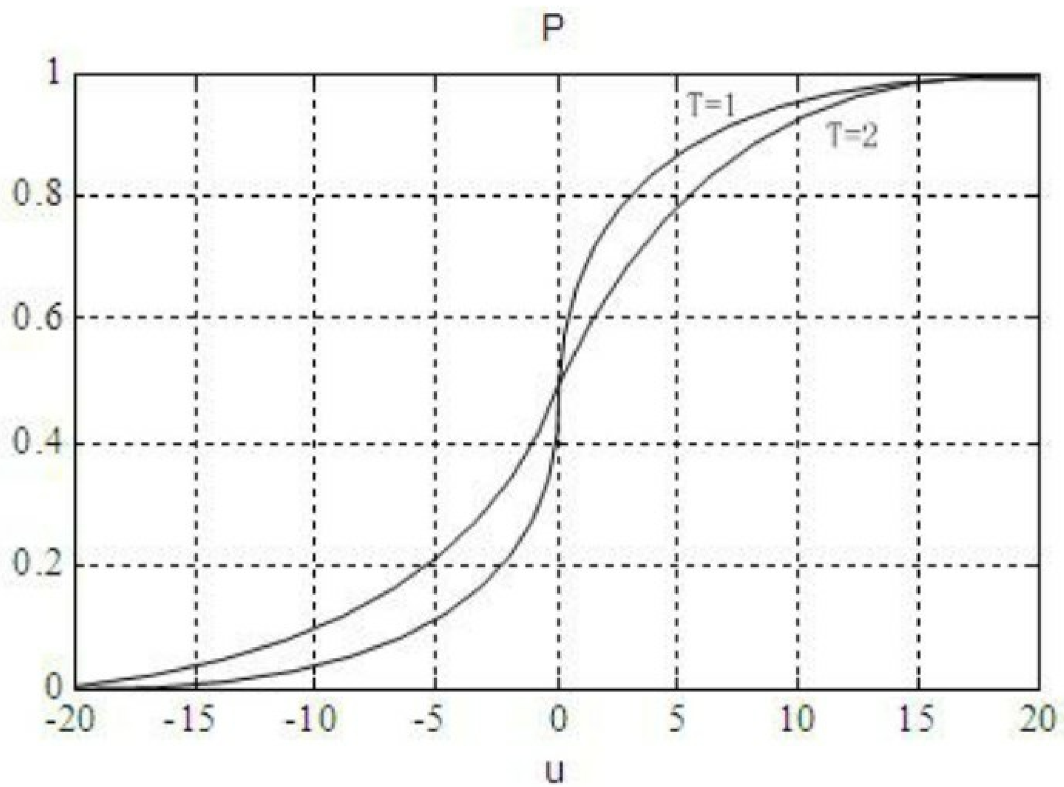


图14-3 P-u之间的关系

可见，T越高时，曲线越平滑，因此，即使u_i有很大变动，也不会对v_i取1的概率变化造成很大的影响；反之，T 越低时，曲线越陡峭，当u_i有稍许变动时就会使概率有很大差异。即温度高时状态变化接近随机，随着温度的降低向确定性的动作靠近。

当$T \rightarrow 0$时，每个神经元不再具有随机特性，而具有确定的特性，激励函数变为阶跃函数，这时Boltzmann 机趋向于Hopfield 网络。

14.3.2 Boltzmann机的工作原理

Boltzmann机采用下式所示的能量函数作为描述其状态的函数。

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} v_i v_j$$

将 Boltzmann 机视为一动力系统，能量函数的极小值对应系统的稳定平衡点，由于能量函数有界，当网络温度以某种方式逐渐下降到某一特定值时，系统必趋于稳定状态， Boltzmann机的运行过程就是逐步降低其能量函数的过程。

Boltzmann 机在运行时，假设每次只改变一个神经元的状态，如第 i 个神经元，设 v_i取0和取1时系统的能量函数分别为0和 $-\sum_j w_{ij} v_j$，它们的差值为ΔE_i。

$$\Delta E_i = E\big|_{v_i=0} - E\big|_{v_i=1} = 0 - \sum_j w_{ij} v_j = \sum_j w_{ij} v_j$$

ΔE_i的取值可能有两种情况：$\Delta E_i>0$或$\Delta E_i<0$。

（1）当$\Delta E_i>0$ 即

$$u_i = \sum_j w_{ij} v_j > 0 \text{时，} \quad E\big|_{v_i=0} > E\big|_{v_i=1}$$

。

神经元取1的概率：$P(v_i = 1) = 1 / (1 + e^{-u_i/T})$

神经元取0的概率：

$$P(v_i = 0) = e^{-u_i/T} / (1 + e^{-u_i/T})$$

当$u_i=\Delta E_i>0$时，$e^{-u_i/T}<1$，$P(v_i=1)>P(v_i=0)$ 这时神经元 i 的状态取 1 的可能性比取0的可能性大，即网络状态取能量低的可能性大。

（2）同理当$\Delta E_i<0$时，即 $u_i=\sum_i w_{ij}v_j<0,$ 那么 $E\big|_{v_i=1}>E\big|_{v_i=0}$。此时$P(v_i=1)<P(v_i=0)$即神经元i的状态取0的可能性比取1的可能性大。

网络状态取能量低的可能性大。运行过程中总的趋势是朝能量下降的方向运动，但也存在能量上升的可能性。

假设有一个凹凸不平的盆，要使一个小球稳定在最低的地方，如果把小球轻轻地放入盆中，那么结果必然是小球稳定在距放入地方最近的低洼处。

但是稳定所在的地方并不保证是最低的地方，这一动作与Hopfield网络相当。Boltzmann机则是大幅度摇晃刚放入小球的盆子，然后逐渐减小摇晃的幅度。这样，小球才有可能到盆子的最低处B处。

假定Boltzmann机中有V_1和V_2两种状态：在V_1状态下神经元i的输出$v_i=1$，V_2状态下神经元 i 的输出 $v_i=0$，而所有其他神经元在这两种状态下的取值都是一致的，另外假设两种状态出现的概率分别是P_{v_1}和P_{v_2}。

$$P_{v_1}=k\cdot P_i=k/(1+e^{-\Delta E_i/T}) \qquad k\text{为常数}$$

$$P_{v_2}=k\cdot(1-P_i)=ke^{-\Delta E_i/T}/(1+e^{-\Delta E_i/T}) \qquad \Delta E_i=E_{v_2}-E_{v_1}$$

对于网络中任意两个状态V_1和V_2的出现概率分别为P_{v1}和P_{v2}。它们之间的关系为

$$\frac{P_{v_1}}{P_{v_2}} = 1 / e^{-\Delta E_i / T} = e^{-(E_{v1} - E_{v2})/T}$$

上式符合统计物理学中已知的Boltzmann分布。Boltzmann机由此得名。

首先，

$$E_{v_1} > E_{v_2} \Rightarrow e^{-(E_{v1} - E_{v2})/T} < 1 \Rightarrow P_{v_1} < P_{v_2}$$

$$E_{v_1} < E_{v_2} \Rightarrow e^{-(E_{v1} - E_{v2})/T} > 1 \Rightarrow P_{v_1} > P_{v_2}$$

这就说明了能量低的状态出现的概率大，能量高的状态出现的概率小。

其次，温度参数T也会影响Boltzmann机处于某种状态的概率。

① T很高时，各状态出现的概率差异大大减小，也就是说网络停留在全局最小点的概率，并不比局部最小点的概率甚至非局部最小点高很多。也即网络不会陷在某个极小点中拔不出来，网络在搜索过程中能够“很快”地穿行于各极小点之间，但落于全局最小点的概率还是最大的。这一点保证网络状态落入全局最小点的可能性大。

② T很低时，情况正好相反。概率差距被加大，一旦网络陷于某个极小点之后，虽然还有可能跳出该极小点，但是所需的搜索次数将是非常多的。这一点保证网络状态一旦达到全局最小点，跳出的可能性就小。

③ $T \to 0$（Hopfield网络)。差距被无限扩展，跳出局部最小点的概率趋于无穷小。这一点保证网络状态稳定在全局最小点。

14.3.3 Boltzmann机的运行步骤

设一个 Boltzmann 机具有 n 个随机神经元（p 个显见神经元，q 个隐见神经元），第 i个神经元与第j个神经元的连接权值为w_{ij}，i，j＝1,2,…,n。T_0为初始温度，m＝1,2,…,M为迭代次数。Boltzmann机的运行步骤如下。

第一步、对网络进行初始化。设定初始温度T_0、终止温度T_{final}和阈值ξ，以及网络各神经元的连接权值w_{ij}。

第二步、在温度T_m条件下（初始温度为T_0）随机选取网络中的一个神经元i，计算神经元i的输入信号总和u_i：

$$u_i = \sum_{\substack{j=1 \\ i \neq j}}^{n} w_{ij} v_j + b_i$$

第三步、若$u_i>0$，即能量差$\Delta E_i>0$，取$v_i=1$为神经元i的下一状态值。若$u_i<0$，计算概率。

$$P_i = 1/(1+e^{-u_i/T})$$

第四步、判断网络在温度T_m下是否达到稳定，若未达到稳定，则继续在网络中随机选取另一神经元j，令j＝i，转至第二步重复计算，直至网络在T_m下达到稳定。若网络在T_m下已达到稳定则转至第五步计算。

第五步、以一定规律降低温度，使$T_{m+1}<T_m$，判断T_{m+1}是否小于T_{final}，若T_{m+1}大于等于T_{final}，则$T_m=T_{m+1}$，转至第二步重复计算；若T_{m+1}小于T_{final}，则运行结束。此时在T_m下所求得的网络稳定状态，即为网络的输出。

Boltzmann机学习需要注意以下几点。

·初始温度 T_0的选择方法。初始温度 T_0的选取主要有以下方法，随

机选取网络中 k个神经元，选取这k个神经元能量的方差作为T_0；在初始网络中选取使ΔE最大的两个神经元，取T_0为ΔE_{max}的若干倍；按经验值给出T_0等。

·确定终止温度阈值 T_{final}的方法。主要根据经验选取，若在连续若干温度下网络状态保持不变，也可认为已达到终止温度。

·概率阈值ξ的确定方法。ξ的选取方法主要有，在网络初始化时按照经验确定或在网络每次运行过程中选取一个[0,0.5]之间均匀分布的随机数。

·网络权值w_{ij}的确定方法。将在下一章节讨论。

·在每一温度下达到热平衡的条件。通常在每一温度下，实验足够多的次数，直至网络状态在此温度下不再发生变化为止。

·降温的方法。通常采用指数的方法进行降温，即

$$T_{m+1} = \frac{T_0}{\log(m+1)}$$

为加快网络收敛速度也可采用倍乘一个小于1的降温系数的方法进行快速降温。

14.3.4 Boltzmann机的学习规则

Boltzmann 机是一种随机神经网络，可使用概率中的似然函数量度其模拟外界环境概率分布的性能。因此，Boltzmann机的学习规则就是根据最大似然规则，通过调整权值w_{ij}，最小化似然函数或其对数。

假设给定需要网络模拟其概率分布的样本集合δ，V_x是样本集合中的一个状态向量， V_x即可代表网络中显见神经元的一个状态，假设向量V_y表示网络中隐见神经元的一个可能状态，则$V=[V_xV_y]$即可表示整个网络所处的状态。

由于网络学习的最终目的是模拟外界给定样本集合的概率分布，而Boltzmann 机含有显见神经元和隐见神经元，因此Boltzmann机的学习过程包括以下两个阶段。

（1）主动阶段。网络在外界环境约束下运行，即由样本集合中的状态向量V_x控制显见神经元的状态。定义神经元i和j的状态在主动阶段的平均关联为

$$\rho_{ij}^{+} = < v_i v_j >^{+} = \sum_{v_x \in \delta} \sum_{v_y} P(V_y \mid V_x) v_i v_j$$

其中概率$P(V_y|V_x)$表示网络的显见神经元约束在V_x下隐见神经元处于V_y的条件概率，它与网络在主动阶段的运行过程有关。

（2）被动阶段。网络不受外界环境约束，显见神经元和隐见神经元自由运行，不受约束。被动阶段的平均关联为：定义神经元i和j的状态在

$$\rho_{ij}^{-} = < v_i v_j >^{-} = \sum_{v_x \in \delta} \sum_{v} P(V) v_i v_j$$

P(V)为网络处于V状态时的概率，v_i和v_j分别是神经元i和j的输出状态。由于网络在自由运行阶段服从Boltzmann分布，因此

$$P(V) = \frac{e^{-E(V)/T}}{\sum_{v} e^{-E(V)/T}}$$

E(V)为网络处于V状态时的能量。

网络的权值w_{ij}需遵循下面的调整规则。

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} = w_{ij}(t) + \frac{\eta}{T}(\rho_{ij}^{+} - \rho_{ij}^{-})$$

$w_{ij}(t)$为在第t步时神经元i，j之间的连接权值，η为学习速率，T是网络温度。

Boltzmann机的优点如下。

（1）通过训练，神经元体现了与周围环境相匹配的概率分布。

（2）网络提供了一种可用于寻找、表示和训练的普遍方法。

（3）若保证学习过程中温度降低得足够慢，根据状态的演化，可以使网络状态的能量达到全局最小点。

但是在Boltzmann机的学习过程中被动阶段的存在具有两个很大的缺点。

（1）增加计算时间。在外界环境约束条件下，一些神经元由外部环境约束，而在自由运行条件下，所有的神经元自由运行，这样增加了Boltzmann机的随机仿真时间。

（2）对于统计错误的敏感。Boltzmann 机的学习规则包含了主动阶段关联和被动阶段关联的差值。当这两种关联相类似时，取样噪声的存在使得这个差值更加不准确。

14.3.5 Boltzmann机的改进

1.确定性Boltzmann机

由于 Boltzmann 机在学习过程中需要计算网络中每对神经元的平均关联。可以证明Boltzmann机的学习时间同网络神经元的数目呈指数关系。Boltzmann机在学习过程中存在的运算时间过长的问题使其很难在实际问题中加以应用。

目前，还没有一种数学方法可以精确评价 Boltzmann 机的行为，但

是可以使用平均场逼近的方法来逼近。在实际中，只需知道网络状态的平均值或网络中神经元状态的平均值即可。

为研究物理学中的Ising或者Sherrington-Kirkpatrick模型，Landau于1937年提出了平均场理论，这是研究连续相变的普遍理论。1985 年D.J.Amit 采用平均场理论研究联想记忆问题。1987年Peterson和Anderson使用平均场理论来研究确定性Boltzmann机。

其方法是基于“考虑来自周围物质的影响时，不是分别考虑各自的影响，而是以全部的平均影响度近似”。

根据平均场理论，首先需要知道网络中每个神经元状态的平均值，令$<v_i>$表示网络中神经元i状态的平均值。神经元i的输出状态以概率规则描述如下。

$$v_i = \begin{cases} +1 & P(u_i) \\ 0 & 1 - P(u_i) \end{cases}$$

$$P(u_i) = \frac{1}{1 + \exp(-u_i / T)}$$

因此，可以用神经元i的输入表达$<v_i>$表示。

$$< v_i >= 1 \cdot P(u_i) + 0 \cdot [1 - P(u_i)] = P(u_i) = 1 / (1 + \exp(-u_i / T))$$

这就是Boltzmann机神经元的平均场逼近。

平均场逼近的基本概念在于，将每个神经元i的真实输入替换为其平均值$<u_i>$，即

$$u_i \approx < u_i > = < \sum_i w_{ij} v_j > = \sum_j w_{ij} < v_j >$$

因此，需要计算由n个神经元组成的Boltzmann机中神经元i的平均输出$<v_i>$。

$$< v_i > = 1/(1+\exp(-u_i/T)) \approx 1/(1+\exp(-< u_i >/T)) = 1/(1+\exp(\sum_j w_{ij} < v_j >))$$

上式表明，一个随机变量函数的平均值可以由此随机变量平均值的函数逼近。

通过平均场逼近可以得到确定的标准Boltzmann学习规则近似为

$$\Delta w_{ij} = \frac{\eta}{T}(U_i^+ U_j^+ - U_i^- U_j^-)$$

其中U_i^+和U_i^-分别是显见神经元i在约束条件和自由运行条件下的平均输出，η为学习速率参数。这种方法称作“确定性Boltzmann学习规则”，而这种神经网络则称作确定性Boltzmann机。

确定性Boltzmann机在实际应用中需注意以下两点。

·确定性Boltzmann 学习规则只在有监督条件下起作用。无监督学习不能在所有的平均场框架中起作用，因为平均状态不能很好地表示自由运行状态的概率分布。

·在有监督学习中，确定性Boltzmann 学习要求神经网络只有一个单隐层（Galland， 1993）。在理论中可使用多个隐层，但在实际中，使用超过一个隐层会导致上述提到的无监督学习的相同问题。

2.Sigmoid置信度网络

针对 Boltzmann 机学习过程中被动阶段存在增加计算时间和对统计错误敏感的缺点， Neal在1992年提出了Sigmoid置信度网络，也称为逻辑推理网络。

提出此网络的目的在于寻找一种随机神经网络，使其可以具有Boltzmann 机从二值向量中学习任意概率分布的能力，而又没有Boltzmann机学习过程中需要被动阶段的缺点。

Sigmoid置信度网络通过控制学习过程而不是使用被动阶段，来避

免上述的缺点。

Sigmoid置信度网络的结构如图14-4所示。

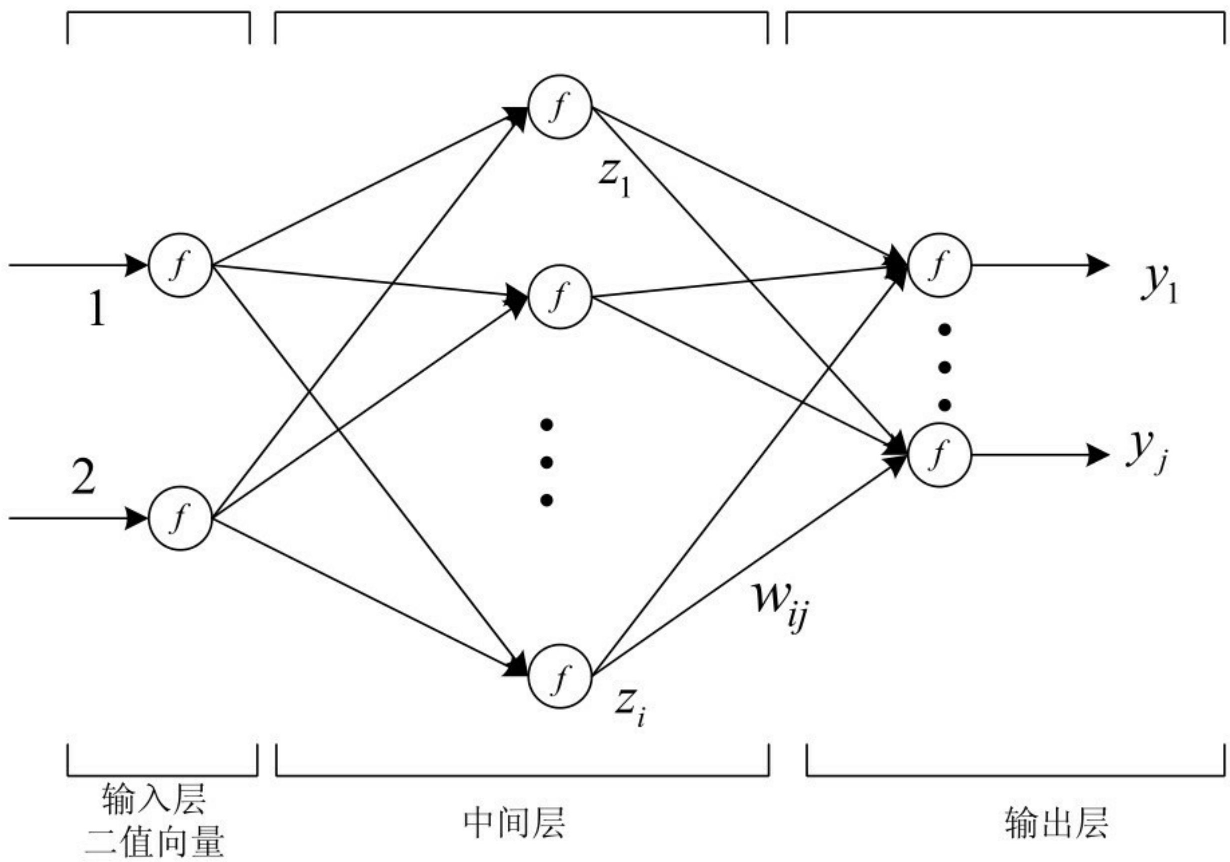


图14-4 Sigmoid置信度网络的结构

Sigmoid置信度网络将Boltzmann机中的对称连接转变为无反馈直接连接的形式，无反馈的连接特性可简化概率计算。Sigmoid置信度网络由多层结构的二值随机神经元构成，并使用Sigmoid函数计算每个神经元的条件概率。

Sigmoid置信度网络结构为典型的前向网络，其输入、输出为二值变量。

设 Sigmoid 置信度网络由 n 个神经元组成，每个神经元的状态由二值随机变量$V_1,V_2,\ldots,V_n$表示，则向量 $V=\{V_1,V_2,\ldots,V_n\}$即可表示网络的状态。定义 $pa(V_i)$为网络中前 i-1个神经元状态的一个子集。

$$pa(V_i) \subseteq \{V_1, V_2, \cdots, V_{i-1}\}$$

$pa(V_i)$是随机向量V的子集。

$$P(V_i = v_i \mid V_1 = v_1, V_2 = v_2, \cdots, V_{i-1} = v_{i-1}) = P(V_i = v_i \mid pa(V_i))$$

第i个神经元的激活概率由Sigmoid函数定义。

$$P(V_i = v_i \mid pa(V_i)) = f\left(\frac{v_i}{T}\sum_{j<i} w_{ij} v_j\right)$$

此处w_{ij}为从神经元j到神经元i的连接权值， f (·)为Sigmoid函数。从上式中可以看出，条件概率 $P(V_i=v_i|pa(V_i))$只与 $pa(V_i)$有关。上式所定义的第 i 个神经元的激活概率是在网络中传播推理的基础。

·对于不属于$pa(V_i)$的所有V_j，$w_{ij}=0$。

·对于所有的，$w_{ij}=0$。

第一点由$pa(V_i)$的定义所决定。第二点是由于Sigmoid置信度网络神经元的直接无反馈连接。Sigmoid置信度网络的随机操作比Boltzmann机要复杂。

这种随机神经网络在概率空间中使用梯度下降的学习算法。

在Sigmoid置信度网络学习算法中，假设每个样本都是二值的。由状态向量V决定网络神经元数量。定义状态向量的子集V_x代表训练数据的特征，即V_x是表示显见神经元的状态向量。剩下的状态向量表示为V_y，即隐见神经元的状态向量。

对于给定状态向量V，Sigmoid置信度网络的设计高度依赖于显见神经元和隐见神经元的排列方式。因此，显见神经元与隐含神经元不同的排列方式会导致不同的结构。

按照梯度下降的思路，将Sigmoid置信度网络神经元的阈值归并至连接权值w_{ij}中。则Sigmoid置信度网络第t+1步的权值调整规则如下。

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$

其中

$$\Delta w_{ij} = \frac{\eta}{T}\rho_{ij}$$

$$\rho_{ij} = \sum_{v_x \in \delta}\sum_{v_y} P(V = v \mid V_x = v_x) f(-\frac{v_i}{T}\sum_{j<i} W_{ij} v_j) v_i v_j$$

神经元i和j的平均关联$w_{ij}(t)$为第t步时神经元i，j的连接权值，η为学习速率，T是网络温度。

Sigmoid 置信度网络在学习过程中摈弃了自由运行的过程，即网络只需在训练样本约束条件下进行学习，从而达到去除被动阶段的目的。

与Boltzmann机不同，在Sigmoid置信网络中只有一个阶段需要学习。这种简化的原因在于，经过Sigmoid函数 f (·) ，状态向量的概率分布在每个神经元的局部水平达到标准化。给定从训练样本集合中抽取的v_x的值，可以正确建模随机向量V的条件分布。

Boltzmann机学习过程中的自由运行阶段由因子

$f(-\frac{v_i}{T}\sum_{j<i} w_{ij} v_j)$ 所取代。

Sigmoid置信度网络将前向网络结构引入随机神经网络的研究中，从而避免Boltzmann机神经元全互联结构增加计算概率时间的缺点。

另一方面，Sigmoid 置信度网络在结构上与前向网络类似，但在前向网络结构的基础上引入了优越的随机机制，从而避免前向网络无法跳出局部极小点等方面的缺点。因此，它结合了随机神经网络与前向神经网络的优点。

14.4 随机神经网络的应用

旅行商问题，即TSP问题（Travelling Salesman Problem）又译为旅行推销员问题、货郎担问题，是数学领域中著名问题之一。假设有一个旅行商人要拜访n个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。

路径的选择目标是要求得的路径路程为所有路径之中的最小值。

TSP问题是一个组合优化问题。该问题可以被证明具有NPC计算复杂性。因此，任何能使该问题的求解得以简化的方法，都将受到高度的评价和关注。

以下利用模拟退火算法解决TSP问题。

【例14-1】随机设置11个城市的坐标，用模拟退火算法解决这些城市的TSP问题，并求解该问题最优解。

解：设置城市坐标为

Coord =[0.6783 0.4195 0.6 0.2949 0.2727 0.4233 0.4174 0.7752 0.6778 0.7488 0.1621;

0.2236 0.1634 0.9439 0.2463 0.1293 0.561 0.6414 0.4536 0.7219 0.4609 0.4781]

根据各城市的距离坐标求相互之间的距离，编写函数如下所示。

```
function [ fare ]= distance( coord )
% 根据各城市的距离坐标求相互之间的距离
% fare为各城市的距离，coord为各城市的坐标
[ n ,m ]= size( coord ) ;            %m为城市的个数
fare= zeros(m ) ;
for i= 1 :m                  %外层为行
```

```
    for j= i :m               %内层为列
        fare( i , j ) = ...
            ( sum( ( coord( : , i ) - coord( : , j ) ) .^ 2 ) ) ^ 0.5 ;
        fare( j , i )= fare( i , j ) ;        %距离矩阵对称
    end
end
```

为求解最优城市路径，对城市位置进行互换操作，编写函数如下所示。

```
function [ newpath , position ] = swap( oldpath , number )
%对oldpath进行互换操作
%number为产生的新路径的个数
%position为对应newpath互换的位置
%城市的个数
m = length( oldpath ) ;
newpath = zeros( number , m ) ;
% 随机产生交换位置
position = sort( randi( m , number , 2 ) , 2 );
for i = 1 : number
    newpath( i , : ) = oldpath ;
%交换路径中选中的城市
    newpath( i , position( i , 1 ) ) = oldpath( position( i , 2 ) ) ;
    newpath( i , position( i , 2 ) ) = oldpath( position( i , 1 ) ) ;
end
```

计算访问城市的顺序，编写函数如下所示。

```
function [ objval ] = pathfare( fare , path )
%path为1到n的排列，代表城市的访问顺序；
%fare为代价矩阵，且为方阵。
```

```
[ m , n ] = size( path ) ;
objval = zeros( 1 , m ) ;
for i = 1 : m
    for j = 2 : n
        objval( i ) = objval( i ) + fare( path( i , j - 1 ) , path( i , j ) ) ;
    end
    objval( i ) = objval( i ) + fare( path( i , n ) , path( i , 1 ) ) ;
end
```

编写访问城市路径图形的函数，如下所示。

```
function [ ] = myplot( path , coord , pathfar )
% 做出路径的图形
% path为要做图的路径，coord为各个城市的坐标
% pathfar为路径path对应的费用
len = length( path ) ;
clf ;
hold on ;
title( [ '近似最短路径如下，费用为' , num2str( pathfar ) ] ) ;
plot( coord( 1 , : ) , coord( 2 , : ) , 'ok');
pause( 0.4 ) ;
for ii = 2 : len
    plot( coord( 1 , path( [ ii - 1 , ii ] ) ) , coord( 2 , path( [ ii - 1 , ii ] ) ) , '-b');
    x = sum( coord( 1 , path( [ ii - 1 , ii ] ) ) ) / 2 ;
    y = sum( coord( 2 , path( [ ii - 1 , ii ] ) ) ) / 2 ;
    text( x , y , [ '(' , num2str( ii - 1 ) , ')' ] ) ;
    pause( 0.4 ) ;
end
```

```
plot( coord( 1 , path( [ 1 , len ] ) ) , coord( 2 , path( [ 1 , len ] ) ) , '-b' ) ;
x = sum( coord( 1 , path( [ 1 , len ] ) ) ) / 2 ;
y = sum( coord( 2 , path( [ 1 , len ] ) ) ) / 2 ;
text( x , y , [ '(' , num2str( len ) , ')' ] ) ;
pause( 0.4 ) ;
hold off ;
```

根据以上函数，编写模拟退火算法程序解决TSP问题，代码如下。

```
clc
clear all;
%设定城市坐标
Coord = ...
  [      0.67830.4195      0.6    0.2949      0.27270.4233      0.41740.7752
0.67780.7488 0.1621; ...0.2236 0.1634 0.9439 0.2463 0.1293 0.561 0.6414
0.4536 0.7219 0.4609 0.4781] ;
t0=1 ;               %初始文图
iLk= 30 ;              %内循环迭代次数
oLk=60 ;               %外循环最大迭代此时
lam = 0.9 ;
istd=0.001 ;           %若内循环函数值方差小于 istd则停止
ostd= 0.001 ;           %若外循环函数值方差小于ostd则停止
ilen= 5 ;              %内循环保存的目标函数值个数
olen= 5 ;              %外循环保存的目标函数值个数
%程序主体
m= length(Coord ) ;        %城市的个数
fare= distance(Coord ) ;        %路径费用
path= 1 :m ;            %初始路径
pathfar= pathfare( fare , path ) ;    %路径费用
```

```
ores= zeros(1 , olen ) ;              %外循环保存的目标函数值
e0=pathfar ;                  %能量初值
t= t0 ;                       %温度
for out= 1 :oLk               %外循环模拟退火过程
    ires= zeros( 1 , ilen ) ;           %内循环保存的目标函数
    for in= 1 : iLk               %内循环模拟热平衡过程
        [ newpath ,～ ]= swap( path , 1 ) ;    %产生新状态
        e1=pathfare( fare , newpath ) ;    %新状态能量
%Metropolis抽样稳定准则
    r = min( 1 , exp( - ( e1 - e0 ) / t ) ) ;
    if rand < r
        path= newpath ;       %更新最佳状态
        e0 = e1 ;
    end
    ires= [ ires( 2 : end ) e0 ] ;       %保存新状态能量
% 内循环终止准则：连续ilen个状态能量波动小于istd
    if std( ires , 1 ) < istd
        break ;
    end
    end
%保存新状态能量
    ores = [ ores( 2 : end ) e0 ] ;
%外循环终止准则：连续olen 个状态能量波动小于ostd
    if std( ores , 1 ) < ostd
        break ;
    end
    t = lam * t ;
```

```
    end
pathfar = e0 ;
%输入结果
fprintf( '近似最优路径是：\n ' )
%disp( char( [ path , path(1) ] + 64 ) ) ;
disp(path)
fprintf( '近似最优路径长度是\tpathfare=' ) ;
disp( pathfar ) ;
myplot( path , Coord , pathfar ) ;
```

运行以上程序，在MATALB 的Command Window窗口显示以下结果。

```
>> N14_1
近似最优路径是：
    9    6    10    8    1    2    5    4    11    7    3
近似最优路径长度是 pathfare=    2.6129
```

运行程序得到的路线图如图14-5所示。

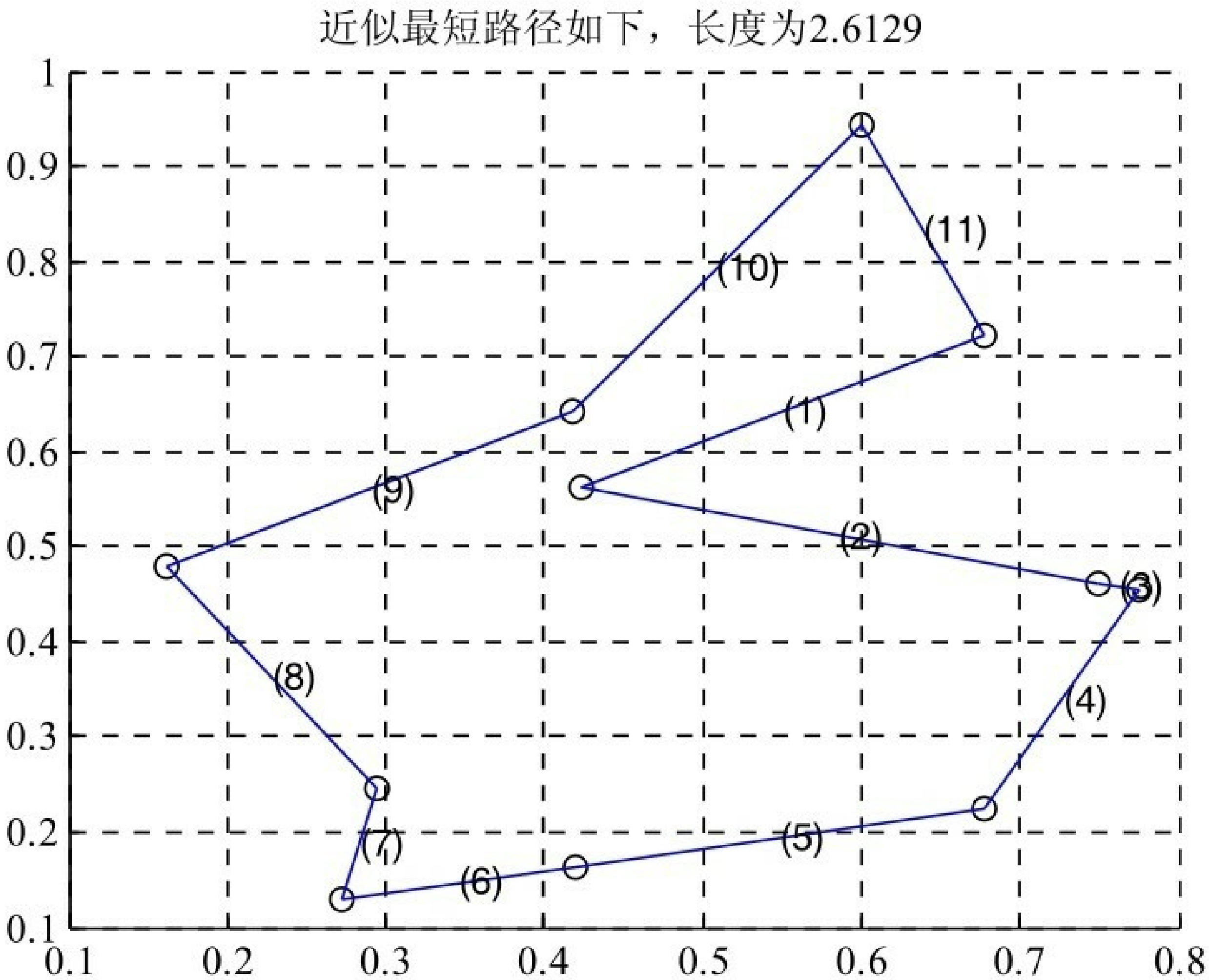


图14-5 运行程序得到的路线图

14.5 本章小结

本章主要介绍了随机神经网络的基本思想、模拟退火算法和Boltzmann机。通过本章介绍可知，尽管模拟退火算法仍然存在着一些不足，但它比快速下降的“贪心”算法的应用型更强。而Boltzmann主要用于优化组合问题。在本章最后运用随机神经网络解决了TSP问题。

第15章 神经网络基础运用

人工神经网络是一种应用类似于大脑神经突触联接的结构进行信息处理的数学模型。在工程与学术界也常直接简称为神经网络或类神经网络。

人工神经网络的基本应用是数据的预测仿真，本章利用简单的感知器神经网络、线性神经网络等说明神经网络的初步应用。

学习目标：

·掌握感知器神经网络的应用

·掌握线性神经网络的应用

·了解BP神经网络的应用

·了解RBF 神经网络的应用

15.1 感知器神经网络的应用

多层感知器神经网络在模式识别、函数逼近、风险预测和控制等领域中有广泛的应用，泛化能力是评价多层感知器神经网络训练成功的重要标准。

多层感知器神经网络从训练样本提取“知识”，实现从输入空间到输出空间的映射，最后用训练过的多层感知器神经网络分类器对新到来的未知样本进行有效的分类。

下面利用感知器神经网络逼近设定的目标。

【例15-1】分别设定一个输入向量和目标向量，利用感知器神经网络完成函数逼近。

解：设定输入向量为P=[-0.7 -0.2 4.2 -0.4 -20;-1 2.7 -0.3 2 40]，目标向量为T=[1 1 1 1 0]，待分类向量的图像如图15-1所示。

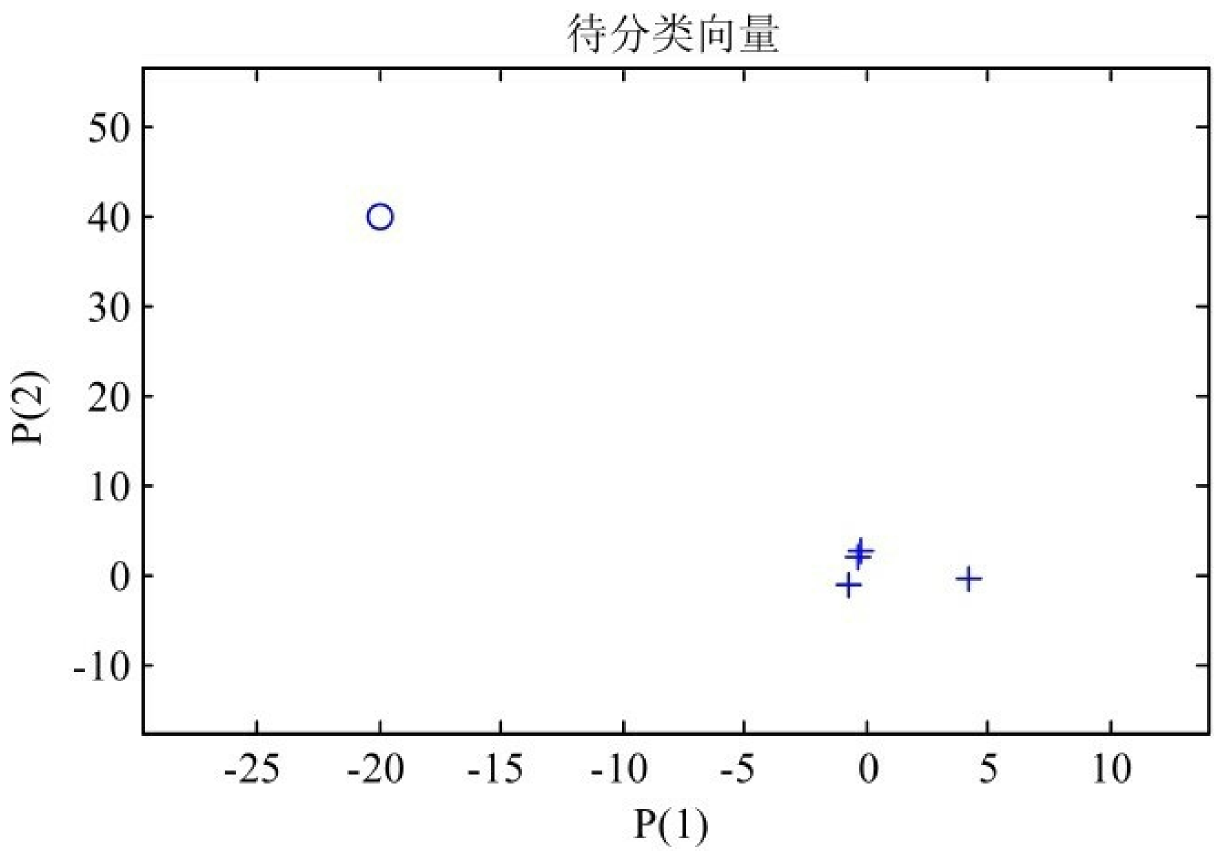


图15-1 待分类向量的图像

建立感知器神经网络的MATLAB代码如下。

```
clear all
clc
P=[-0.7 -0.2 4.2 -0.4 -20;-12.7 -0.3 240];    %输入向量
T=[1 11 10];                 %目标向量
plotpv(P,T);                 %描绘输入点图像
net=newp([-20   4.2;-1   40],1);%生成网络，其中参数分别为输入向量的范围和神经元感应器数量
hold on
linehandle=plotpc(net.iw{1},net.b{1});
net.adaptparam.passes=3;
for a=1:30                %训练次数
```

```
[net,Y,E]=adapt(net,P,T);
linehandle=plotpc(net.iw{1},net.b{1},linehandle);
drawnow;
end
```

运行程序得到结果如图15-2所示。

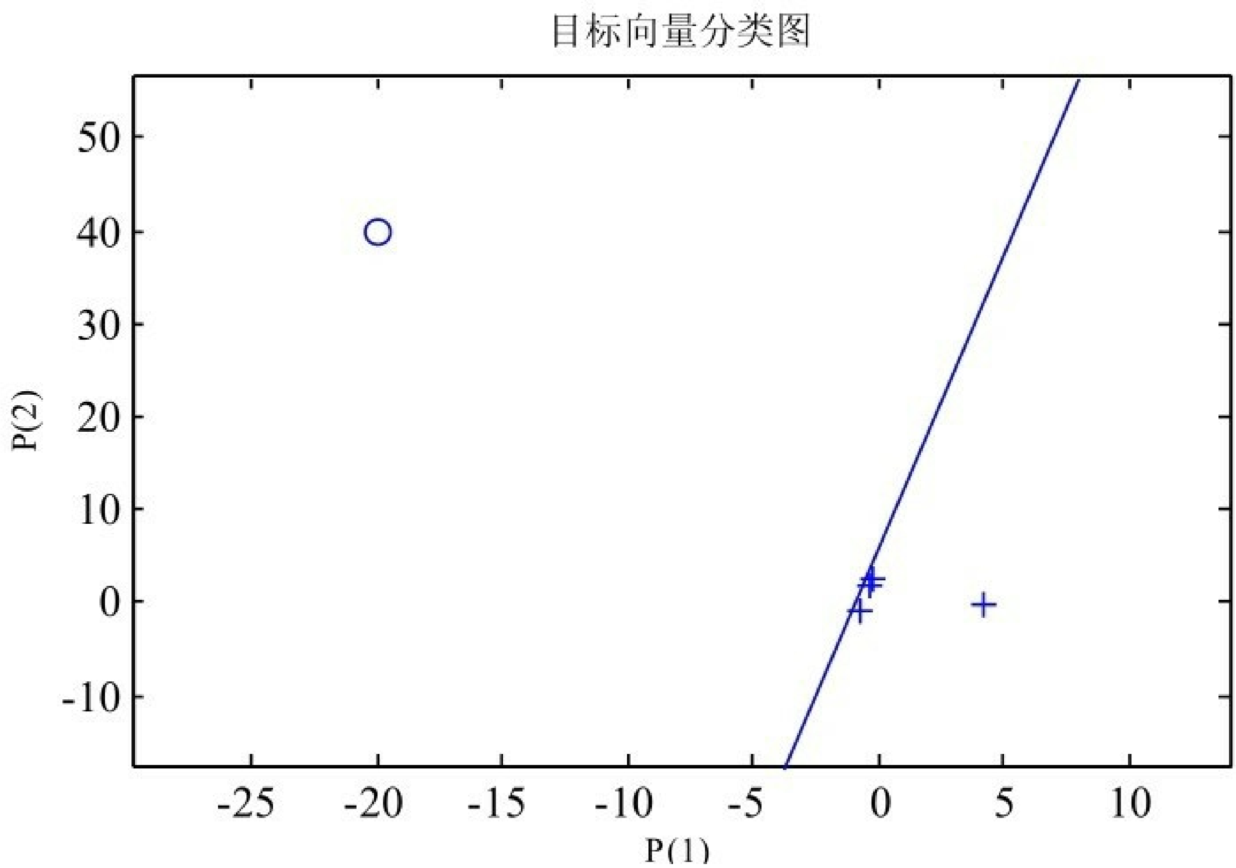


图15-2 目标向量分类图

15.2 线性神经网络的应用

除了神经元的传递函数不同，线性神经网络与感知器网络非常相似。单层线性神经网络和感知器网络一样，都只能解决线性可分的问题。线性神经网络的学习算法比感知器网络的学习算法的收敛速度和精度都有较大的提高。

本节利用线性神经网络完成函数逼近。

【例15-2】已知函数y=sin(3xπ)，利用线性神经网络进行函数逼近。

解：首先建立以下代码。

```
x=0:0.02:5;
T=sin(x*3*pi);
```

运行后得到待预测信号图形如图15-3所示。

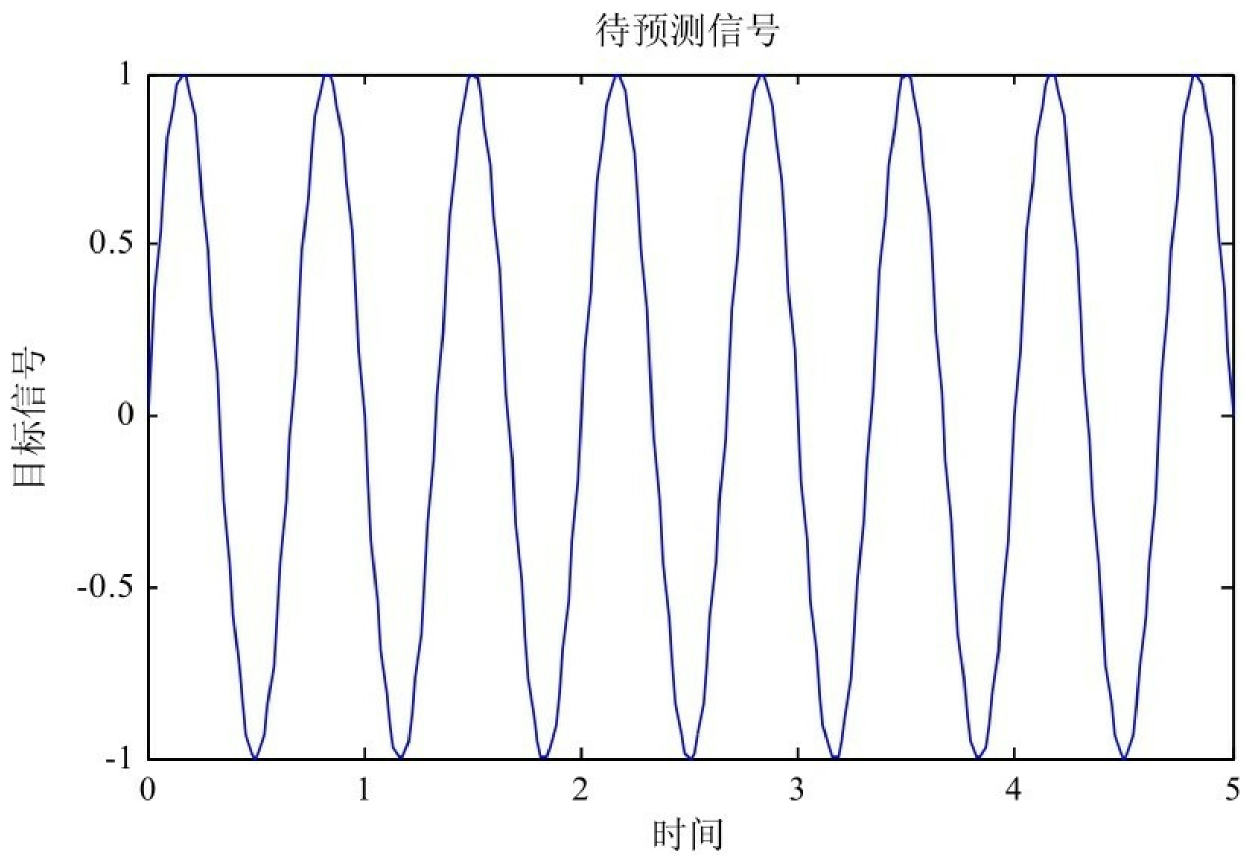


图15-3 待测信号图形

最终建立线性神经网络的代码如下所示。

```
clear all
clc
x=0:0.02:5;
T=sin(x*3*pi);
Q=length(T);
P=zeros(4,Q);    %P中存储信号T的前 4次值，作为网络输入
P(1,2:Q)=T(1,1:(Q-1));
P(2,3:Q)=T(1,1:(Q-2));
P(3,4:Q)=T(1,1:(Q-3));
P(4,5:Q)=T(1,1:(Q-4));
plot(x,T)       %绘制信号T曲线
```

```
xlabel('时间');
ylabel('目标信号');
title('待预测信号');
net=newlind(P,T);    %根据输入和期望输出直接生成线性网络
a=sim(net,P);    %网络测试
figure(2)
plot(x,a,x,T,'+')
xlabel('时间');
ylabel('仿真和目标');
title('仿真信号和目标信号');
e=T-a;
figure(3)
plot(x,e)
hold on
plot([min(x) max(x)],[0 0],'r:')
hold off
xlabel('时间');
ylabel('误差');
title('误差信号');
```

运行后得到仿真信号和目标信号比较图如图15-4所示。

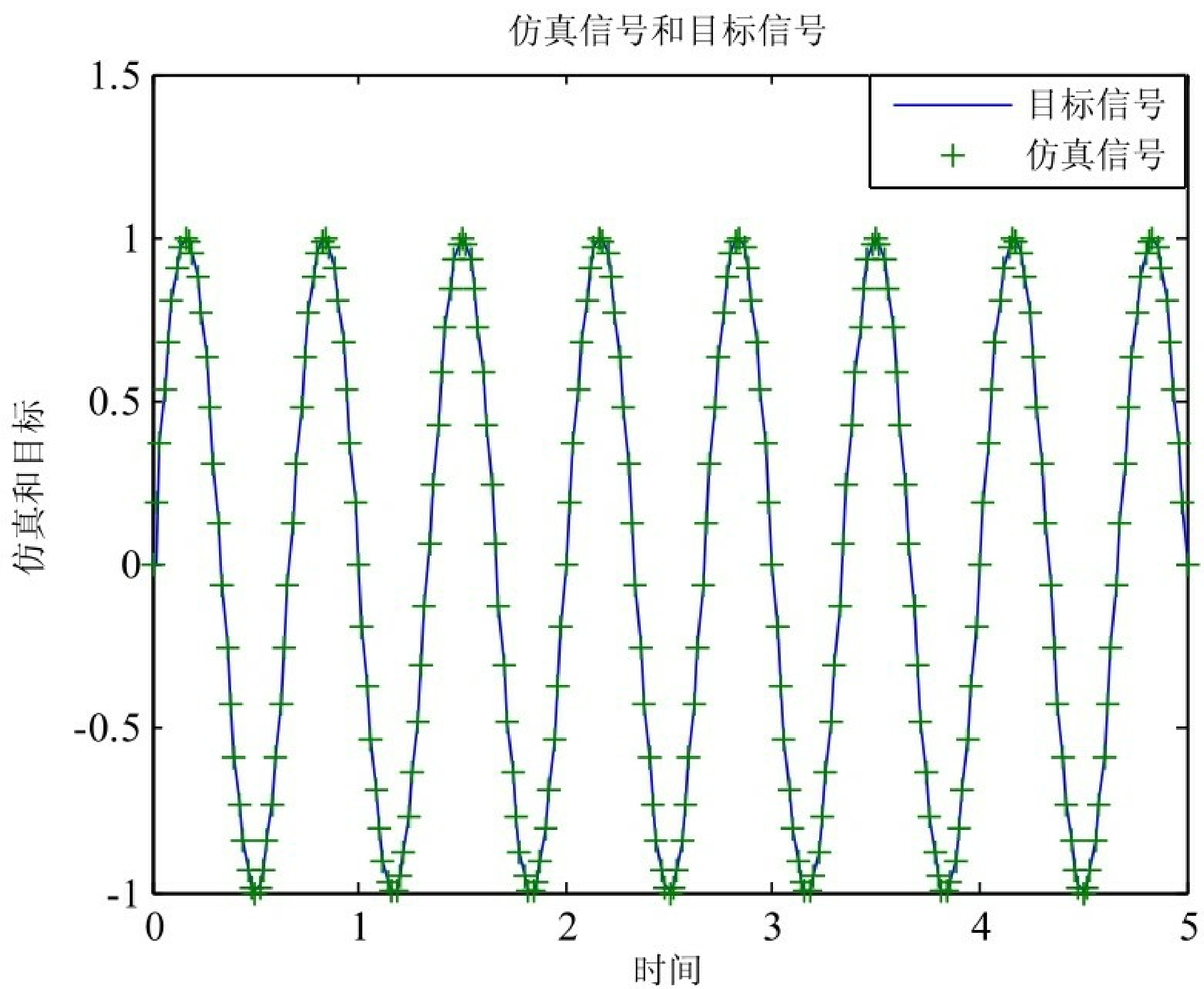


图15-4 仿真信号和目标信号比较图

从图15-4中可以看出，仿真信号和目标信号基本重合，说明建立的线性神经网络的函数逼近能力良好。

15.3 BP神经网络的应用

BP神经网络是1986年由Rumelhart和McCelland为首的科学家小组提出的，是一种按误差逆传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型之一。

BP神经网络常用的网络建立函数如下。

net = newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF)

其中，BTF 表示神经网络的训练函数。在建立 BP 神经网络的过程中，常用的训练函数有以下11种。

·训练函数traingd——梯度下降法，有7 个训练参数。

·训练函数traingdm——有动量的梯度下降法，附加1 个训练参数mc（动量因子，缺省为0.9）。

·训练函数traingda——有自适应lr的梯度下降法，附加3个训练参数：lr_inc（学习率增长比，缺省为1.05。

·训练函数 traingdx——有动量的梯度下降法中赋以自适应 lr 的方法，附加 traingdm和traingda的4个附加参数。

·训练函数trainrp——弹性梯度下降法，可以消除输入数值很大或很小时的误差，附加4个训练参数。

·训练函数traincgf——Fletcher-Reeves共轭梯度法；训练函数traincgp——Polak-Ribiere共轭梯度法。

·训练函数traincgb——Powell-Beale共轭梯度法。共轭梯度法占用存储空间小，附加1训练参数searchFcn（一维线性搜索方法，缺省为srchcha）。

·训练函数trainscg——量化共轭梯度法。

·训练函数trainbfg——BFGS 拟牛顿回退法，收敛速度快，但需要

更多内存，与共轭梯度法训练参数相同。

·训练函数trainoss——一步正割的BP 训练法，解决了BFGS 消耗内存的问题，与共轭梯度法训练参数相同。

·训练函数trainlm——Levenberg-Marquardt 训练法。

【例15-3】任意设置输入和目标向量，建立BP神经网络，使得输入向量的仿真数据与目标向量一致。

解：任意设置的输入P和目标向量t如下所示。

```
P=[-1 -2 1 2;0 4 0 4];
t=[-1 -1 -1 1];
```

利用梯度下降法建立BP神经网络进行数据预测，建立的MATLAB代码如下。

```
clear all
clc
P=[-1 -2 1 2;0 4 0 4];
t=[-1 -1 -1 1];
net=newff(minmax(P),[3,1],{'tansig','purelin'},'traingd');
net=init(net);
net.trainparam.epochs=300;   %最大训练次数(前缺省为10,自   trainrp后，缺省为100)
net.trainparam.lr=0.05;      %学习率(缺省为0.01)
net.trainparam.show=50;   %限时训练迭代过程(NaN表示不显示，缺省为25)
net.trainparam.goal=1e-5;   %训练要求精度(缺省为0)
net.trainparam.max_fail   %最大失败次数(缺省为5)
net.trainparam.min_grad   %最小梯度要求(前缺省为1e-10，自trainrp后，缺省为1e-6)
net.trainparam.time      %最大训练时间(缺省为 inf)
```

```
[net,tr]=train(net,P,t);      %网络训练
a=sim(net,P)          %网络仿真
%绘制仿真结果图形
x=1:4;
plot(x,t,x,a,'r*');
title('仿真结果');
xlabel('输入向量P');
ylabel('目标向量T');
```

运行以上代码，得到BP神经网络训练结果如图15-5所示，在图15-5上方是建立BP神经网络的结构。BP神经网络训练误差曲线如图15-6所示。

在图 15-6 中可以看出神经网络经过 300 步训练，其网络仿真误差达到 0.000 131 94。其仿真结果如图15-7所示。

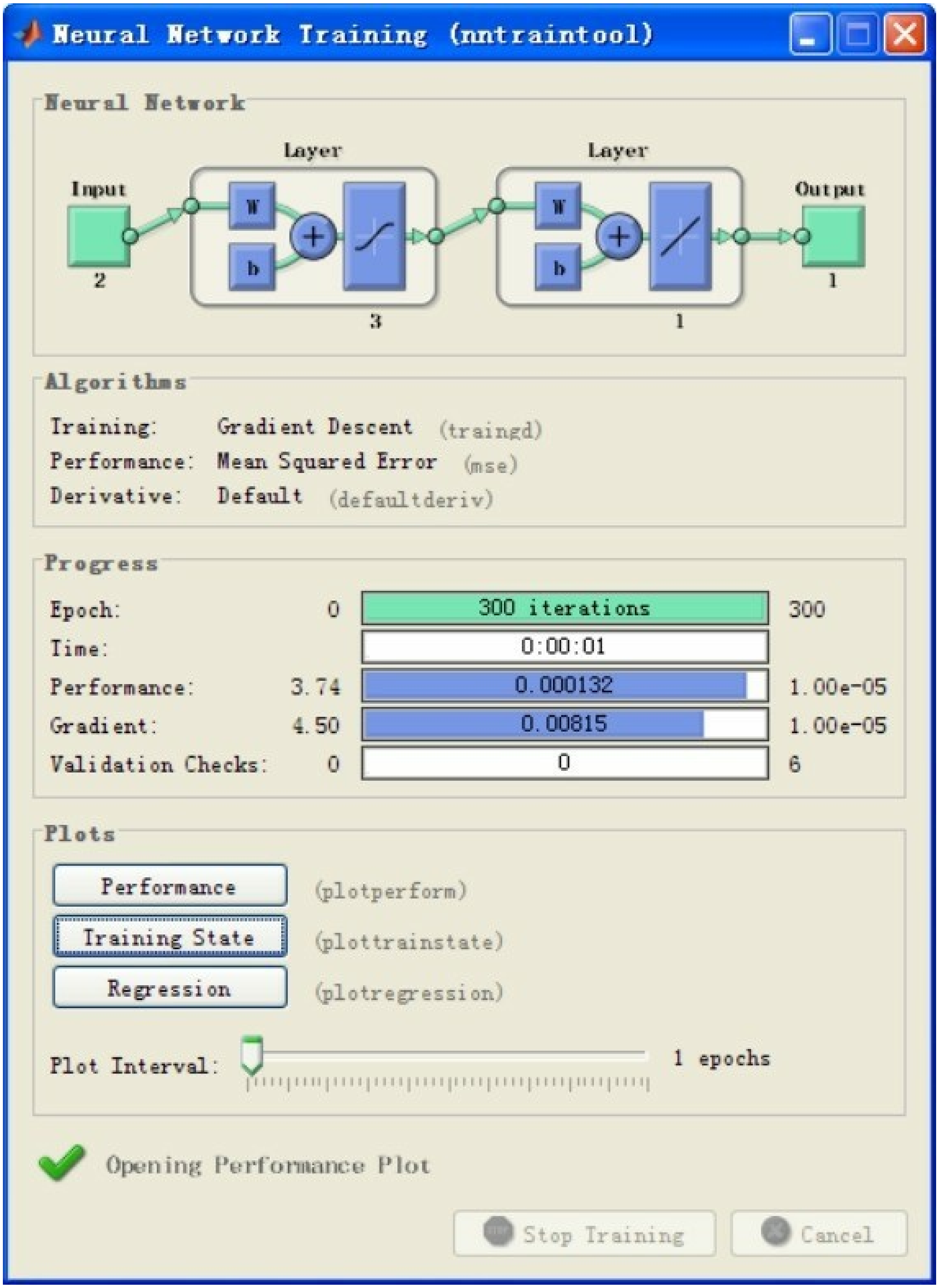


图15-5 BP神经网络训练结果

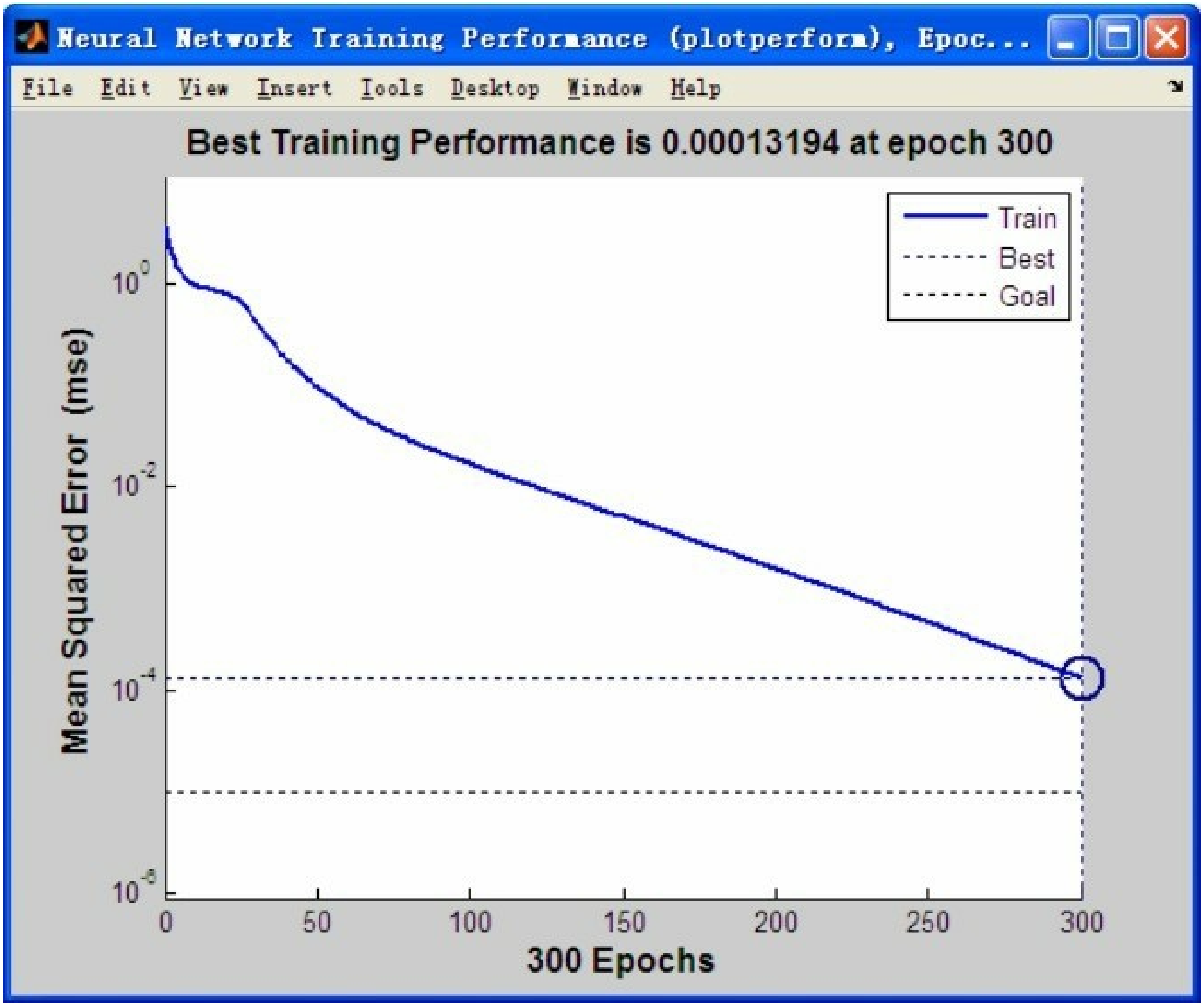


图15-6 BP神经网络训练误差曲线

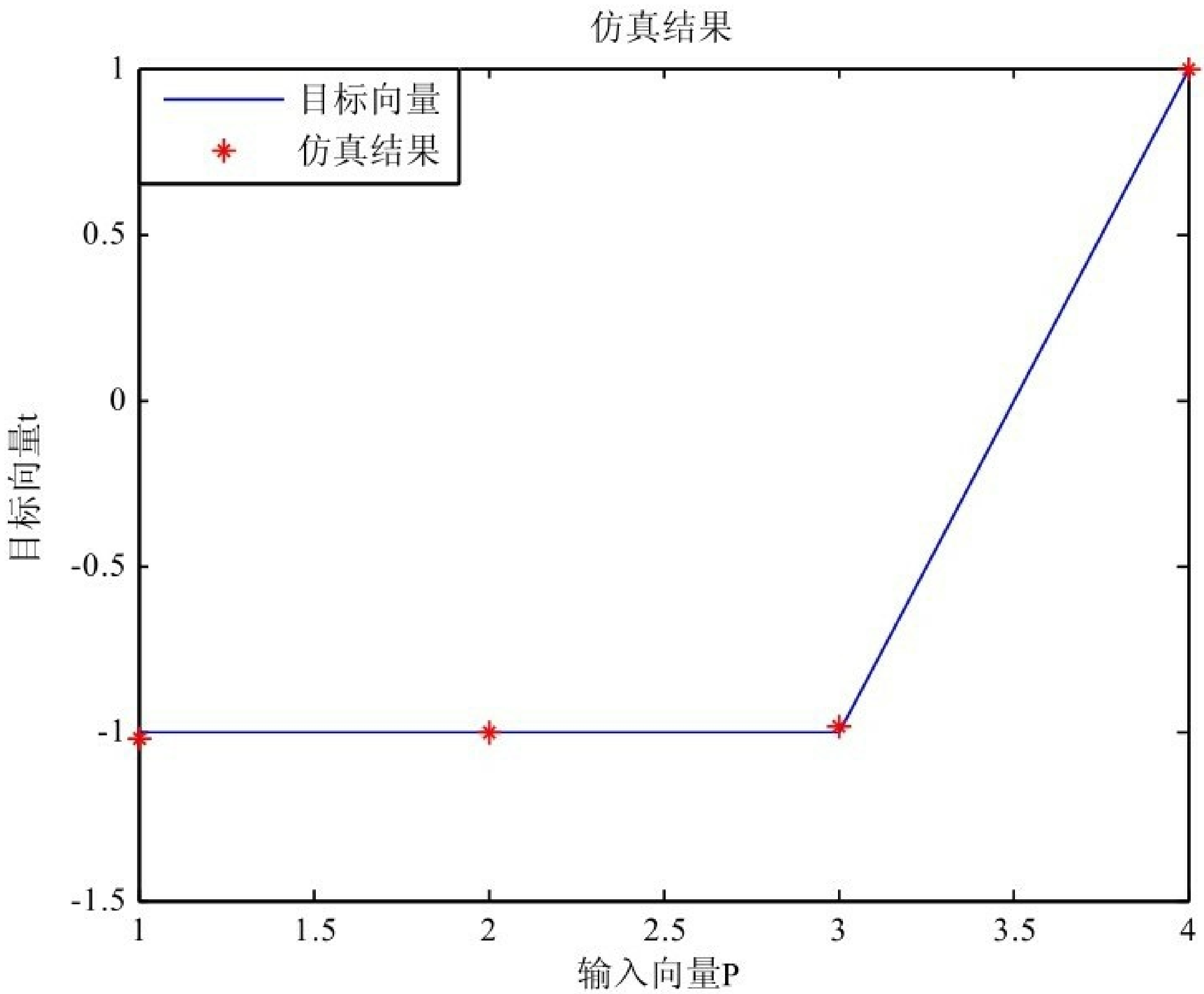


图15-7 仿真结果图形

15.4 RBF神经网络的应用

通用RBF神经网络的逼近能力、分类能力、学习速度方面均优于BP神经网络。在径向基网络中，径向基层的散布常数是spread的选取是关键。spread越大，需要的神经元越少，但精度会相应下降。spread的缺省值为1。

在RBF神经网络的应用中，可以通过net=newrbe(P,T,spread)生成网络，且误差为0；可以通过net=newrb(P,T,goal,spread)生成网络，神经元由1开始增加，直到达到训练精度或神经元数目最多为止。

【例15-4】利用newgrnn函数建立RBF神经网络，并利用设定的输入向量检验RBF神经网络的数据逼近能力。

解：设计一个径向基函数网络，网络有两层，隐层为径向基神经元，输出层为线性神经元。建立神经网络的MATLAB代码如下。

```
clear all
clc
P=[4 5 6];
T=[1.5 3.6 6.7];
net=newgrnn(P,T);
%仿真验证
p=4.5;
v=sim(net,p)
%PNN网络,概率神经网络
P=[0 0 ;1 1;0 3;1 4;3 1;4 1;4 3]';
Tc=[1 1 2 2 3 3 3];
%将期望输出通过ind2vec()转换,并设计、验证网络
```

T=ind2vec(Tc);

net=newpnn(P,T);

Y=sim(net,P);

Yc=vec2ind(Y)

%尝试用其他的输入向量验证网络

P2=[1 4;0 1;5 2]';

Y=sim(net,P2);

Yc=vec2ind(Y)

%应用newrb()函数构建径向基网络,对一系列数据点进行函数逼近

P=-1:0.1:1;

T=[-0.9602 -0.5770 -0.0729 0.3771 0.6405 0.6600 0.4609...

0.1336 -0.2013 -0.4344 -0.500 -0.3930 -0.1647 -0.0988...

0.3072 0.3960 0.3449 0.1816 -0.0312 -0.2189 -0.3201];

%绘制训练用样本的数据点

plot(P,T,'r*');

title('训练样本');

xlabel('输入向量P');

ylabel('目标向量T');

%设计一个径向基函数网络，网络有两层,隐层为径向基神经元,输出层为线性神经元

%绘制隐层神经元径向基传递函数的曲线

p=-3:.1:3;

a=radbas(p);

plot(p,a)

title('径向基传递函数')

xlabel('输入向量p')

%应用newrb()函数构建径向基网络的时候,可以预先设定均方差精

度eg以及散布常数sc

eg=0.02;

sc=1; %其值的选取与最终网络的效果有很大关系,过小造成过适性,过大造成重叠性

net=newrb(P,T,eg,sc);

%网络测试

plot(P,T,'*')

xlabel('输入向量');

X=-1:.01:1;

Y=sim(net,X);

hold on

plot(X,Y);

hold on

legend('目标向量','仿真输出')

运行代码得到建立的RBF神经网络仿真结果如图15-8所示。

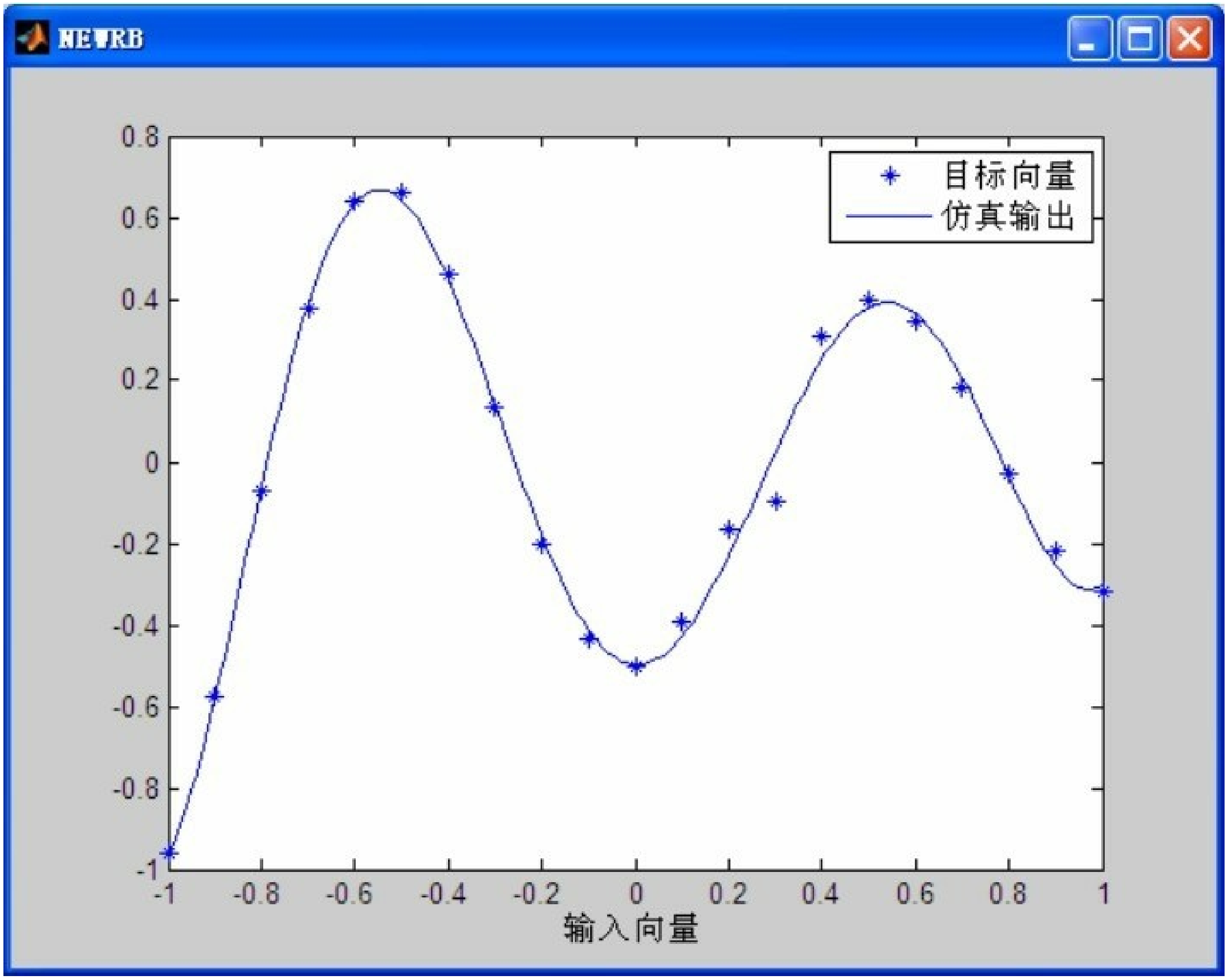


图15-8 RBF神经网络仿真结果

在MATLAB的Command Window中输入如下代码。

```
>> e=sum(P-T)/3
```

可以得到RBF神经网络仿真的平均误差如下。

```
e =
   0.1569
```

RBF神经网络平均误差说明建立的RBF神经网络性能良好。

15.5 本章小结

人工神经网络的构筑理念是受到生物（人或其他动物）神经网络功能的运作启发而产生的。人工神经网络通常是通过一个基于数学统计学类型的学习方法（Learning Method）得以优化，所以人工神经网络也是数学统计学方法的一种实际应用。

本章利用典型案例，讲解了感知器神经网络、线性神经网络、BP神经网络和RBF神经网络的简单应用。

第16章 神经网络综合运用

在传统的非线性系统辨识的理论研究和实际应用方面，采用基于非线性自回归滑动平均模型进行预测存在着极大的困难。相比之下，神经网络在这方面显示了极大的优越性。不仅如此，神经网络在数据拟合、故障诊断等方面，也有很广泛的应用。

本章以几个神经网络的运用为示例，介绍如何使用神经网络工具箱，实现基于神经网络的预测、拟合等。

学习目标：

·掌握BP神经网络的使用方法

·掌握PID 神经网络控制方法

·了解遗传算法的使用

·掌握模块神经网络控制

·了解概率神经网络分类预测方法

16.1 BP神经网络的应用

BP算法是一个适合多层网络的学习算法，由此算法训练的神经网络称之为BP神经。BP网络广泛应用于函数逼近、模式识别/分类、数据压缩等，下面从几个方面介绍BP神经网络的应用。

16.1.1 数据拟合

【例16-1】采用两种训练方法，即L-M优化算法（trainlm）和贝叶斯正则化算法（trainbr），用以训练BP网络，使其能够拟合某一附加有白噪声的正弦样本数据。

解：假定样本数据的输入矢量：P = [-1:0.05:1]；

目标矢量：randn(’seed’,78341223)；

T = sin(2·pi·P)+0.1·randn(size(P))；

解：本例的 MATLAB 程序如下。

1.采用L-M优化算法

```
clear all
clc
% 定义训练样本变量
% P 为输入变量
P = [-1:0.05:1];
% T 为目标变量
%randn('seed',78341223);
T = sin(2*pi*P)+0.1*randn(size(P));
%绘制样本数据点
```

```
figure(1)
plot(P,T,'+');
title('绘制样本数据点');
xlabel('输入变量');
ylabel('目标变量');
hold on;
```

得到样本数据点图形如图16-1所示。

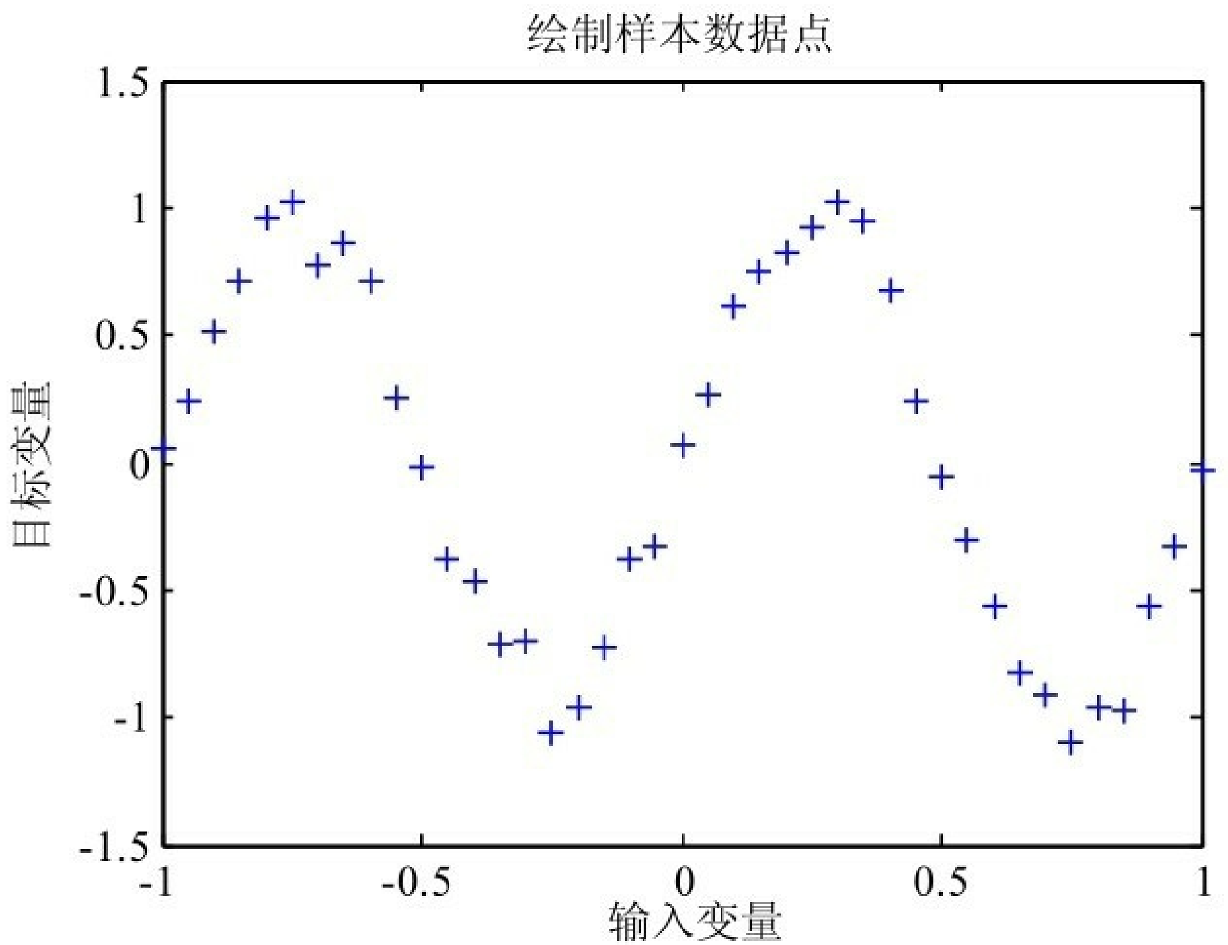


图16-1 绘制样本数据点图形

```
% 绘制不含噪声的正弦曲线
figure(2)
plot(P,sin(2*pi*P),':');
```

```
title('绘制不含噪声的数据点');
xlabel('输入变量');
ylabel('不含噪声的数据点');
hold on
```

得到不含噪声的样本数据点图形如图16-2所示。

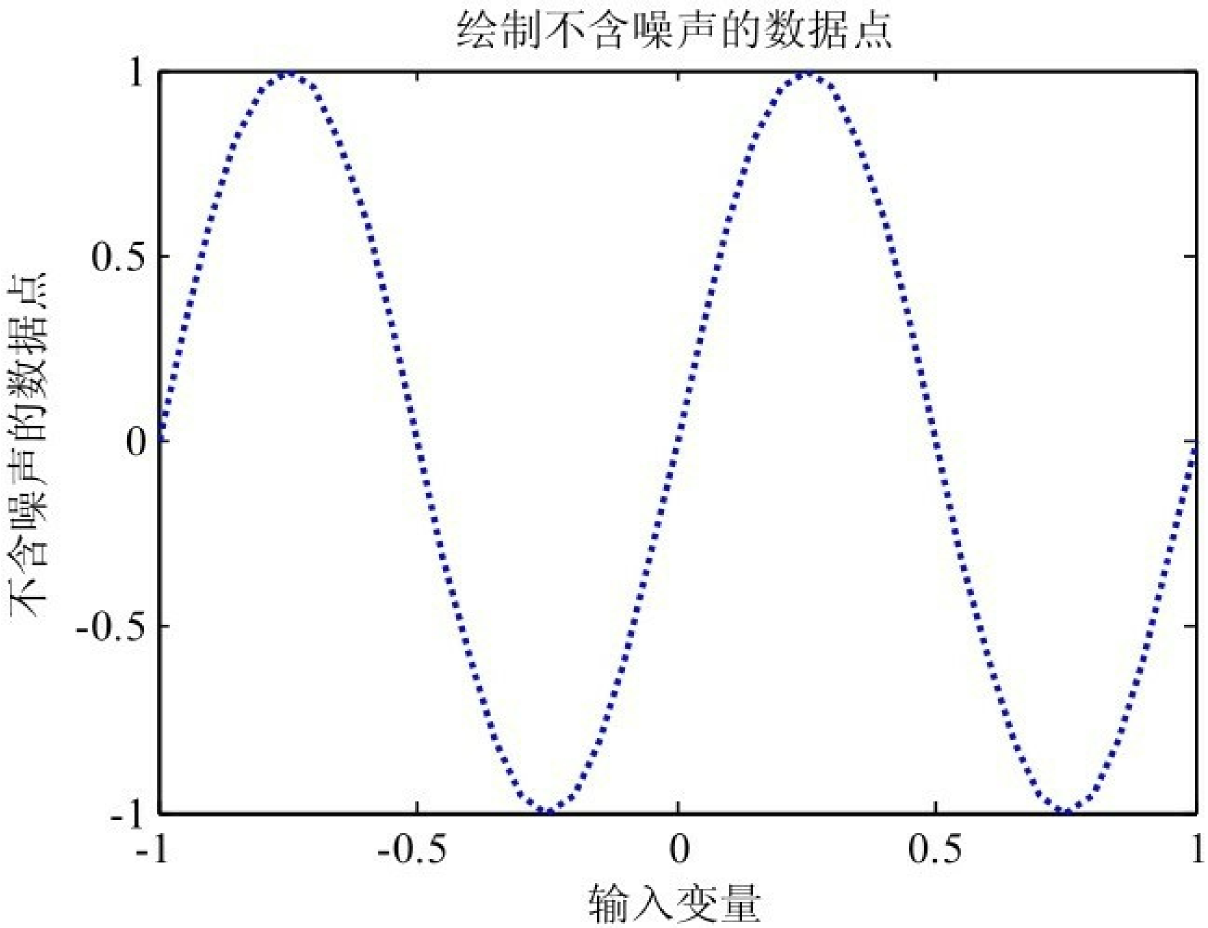


图16-2 绘制不含噪声的样本数据点图形

从图16-1和图16-2可以看出，加入噪声后，标准的正弦波形数据点变得有些没有规律。

```
% 创建一个新的前向神经网络
net=newff(minmax(P),[20,1],{'tansig','purelin'});
% 采用 L-M 优化算法 TRAINLM
net.trainFcn='trainlm';
```

```
% 设置训练参数
net.trainParam.epochs = 200;
net.trainParam.goal = 1e-6;
% 重新初始化
net=init(net);
% 调用相应算法训练 BP 网络
[net,tr]=train(net,P,T);
% 对 BP 网络进行仿真
A = sim(net,P);
% 计算仿真误差
E = T - A;
MSE=mse(E)
% 绘制匹配结果曲线
figure(3)
plot(P,A,P,T,'+');
title('仿真结果比较');
xlabel('输入变量');
ylabel('仿真误差和目标变量');
hold on
```

训练得到的神经网络仿真输入结果曲线如图16-3所示。

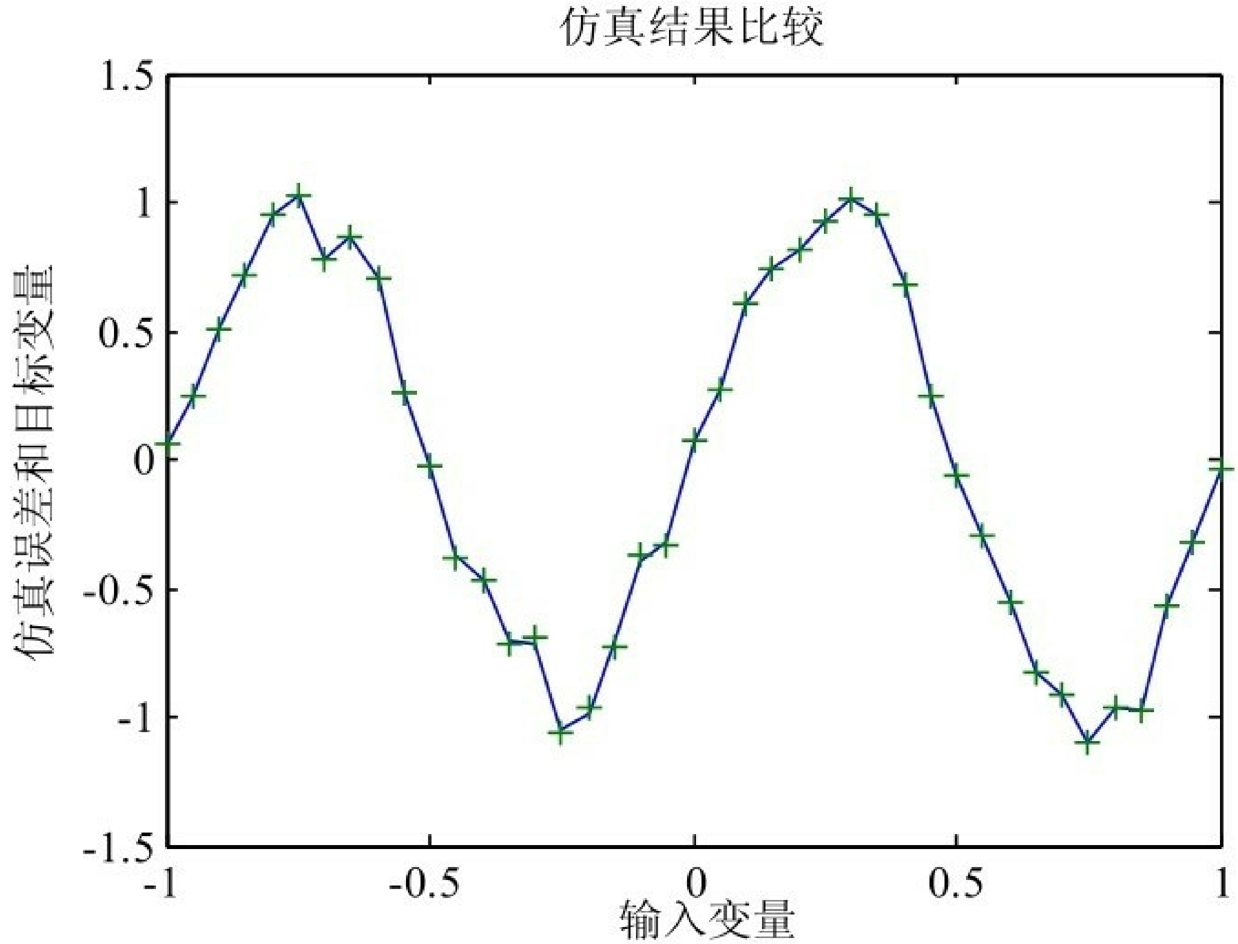


图16-3 采用L-M优化算法的仿真结果比较

训练BP网络时，神经网络的输出误差变化图如图16-4所示。神经网络训练过程中的输出误差达到了设定的目标值。

训练得到的神经网络仿真所得均方误差如下。

```
>> MSE
MSE =
   1.5833e-05
```

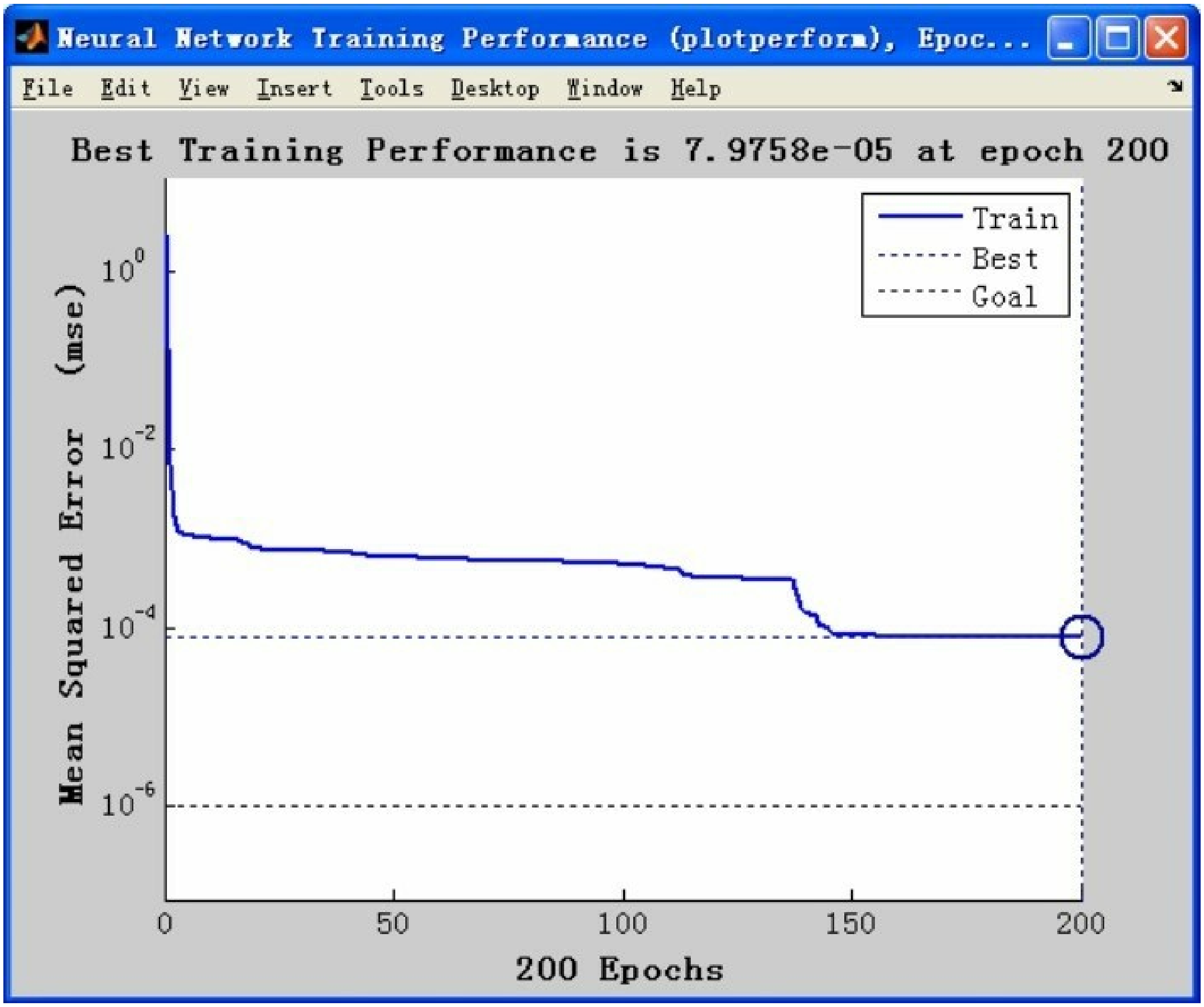


图16-4 神经网络的输出误差变化图

2.采用贝叶斯正则化算法

```
clear all
clc
% 定义训练样本变量
% P 为输入变量
P = [-1:0.05:1];
% T 为目标变量
%randn('seed',78341223);
T = sin(2*pi*P)+0.1*randn(size(P));
```

```
% 绘制样本数据点
figure(1)
plot(P,T,'+');
title('绘制样本数据点');
xlabel('输入变量');
ylabel('目标变量');
hold on;
% 绘制不含噪声的正弦曲线
figure(2)
plot(P,sin(2*pi*P),':');
title('绘制不含噪声的数据点');
xlabel('输入变量');
ylabel('不含噪声的数据点');
hold on
% 创建一个新的前向神经网络
net=newff(minmax(P),[20,1],{'tansig','purelin'});
% 采用贝叶斯正则化算法 TRAINBR
net.trainFcn='trainbr';
% 设置训练参数
net.trainParam.epochs = 200;
net.trainParam.goal = 1e-6;
% 重新初始化
net=init(net);
%% 采用贝叶斯正则化算法 TRAINBR
% net.trainFcn='trainbr';
%% 设置训练参数
% net.trainParam.epochs = 200;
```

```
% randn('seed',192736547);
%% 重新初始化
% net = init(net);
%
% 调用相应算法训练 BP 网络
[net,tr]=train(net,P,T);
% 对 BP 网络进行仿真
A = sim(net,P);
% 计算仿真误差
E = T - A;
MSE=mse(E)
% 绘制匹配结果曲线
figure(3)
plot(P,A,P,T,'+');
title('仿真结果比较');
xlabel('输入变量');
ylabel('仿真误差和目标变量');
hold on
```

训练得到的神经网络仿真输入结果曲线如图16-5所示。

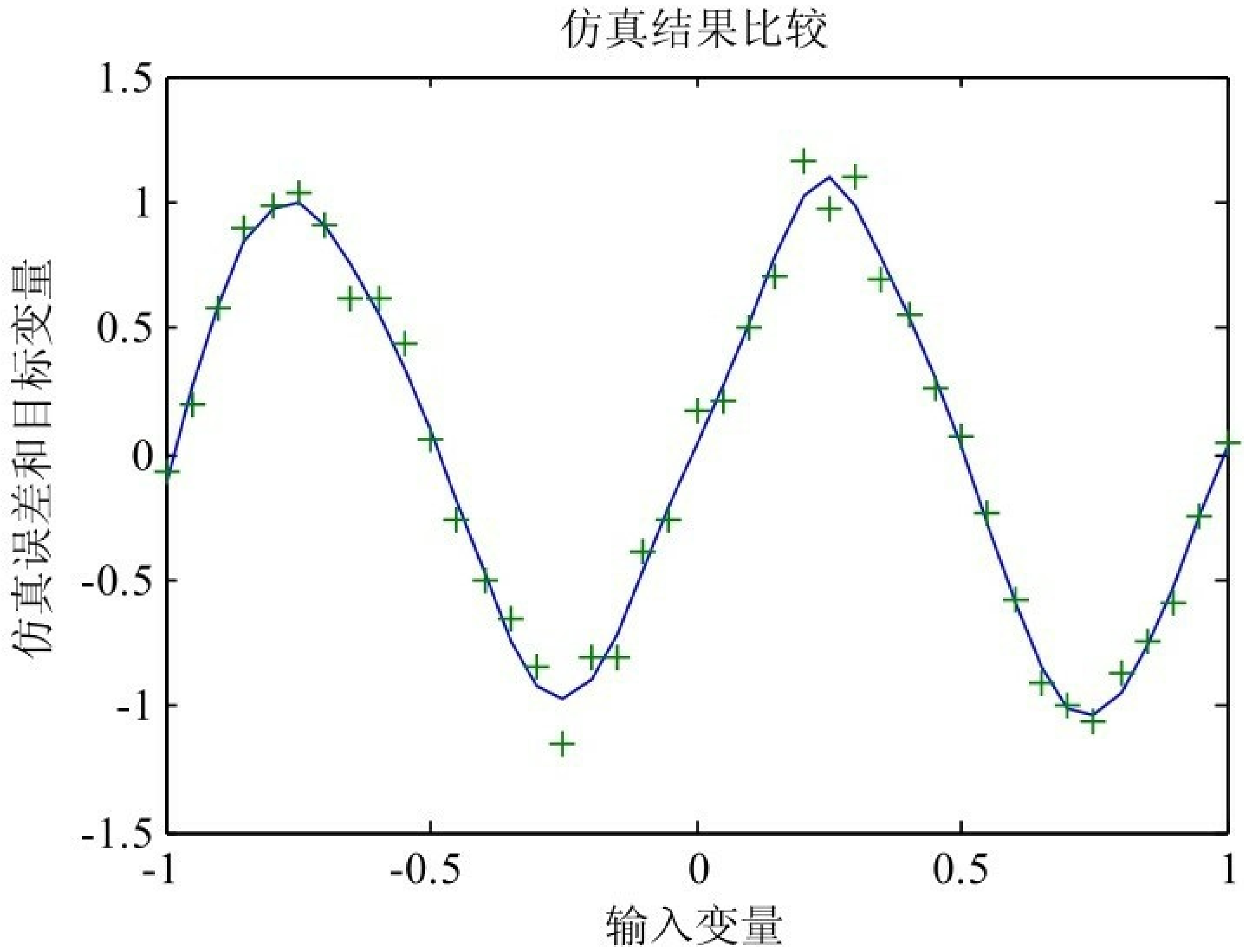


图16-5 采用贝叶斯正则化算法的仿真结果比较

训练BP网络时，神经网络的输出误差变化图如图16-6所示。从图中可以看出，训练过程中网络的输出误差没有达到设定的误差目标。网络误差在训练过程中的变化趋势为先变小后变大，这表明网络存在过度训练的问题。

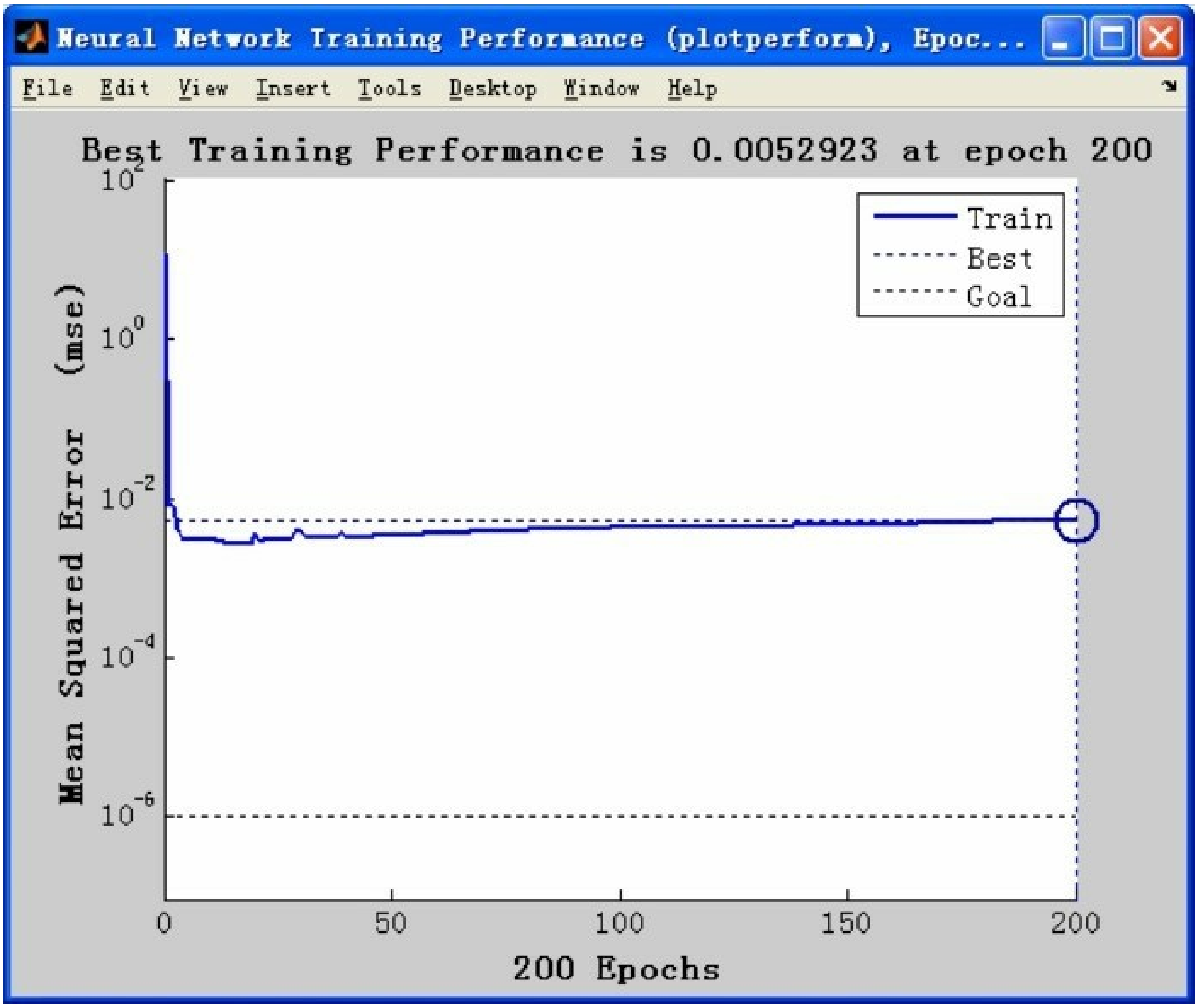


图16-6 神经网络的输出误差变化图

训练得到的神经网络仿真所得均方误差如下。

```
>> MSE
MSE =
   0.0053
```

由以上仿真结果可知，使用L-M优化算法的仿真结果要好于贝叶斯正则化算法的仿真结果。经trainlm函数训练后的神经网络对样本数据点实现了“过度匹配”，而经trainbr函数训练的神经网络对噪声不敏感，具有较好的推广能力。

如果想要提高上述例题中BP神经网络的推广能力，可以采用训练

函数traingdx和“提前停止”相结合的方法来训练BP网络。

在利用“提前停止”方法时，首先应分别定义训练样本、验证样本或测试样本，其中，验证样本是必不可少的。

定义并使用验证样本如下所示。

验证样本输入矢量：vP = [-0.975:0.05:0.975]

验证样本目标矢量：vT = sin(2·pi·val.P)+0.1·randn(size(val.P))

值得注意的是，尽管“提前停止”方法可以和任何一种 BP 网络训练函数一起使用，但是不适合同训练速度过快的算法联合使用，比如 trainlm 函数。

下面采用训练速度相对较慢的变学习速率算法traingdx函数作为训练函数。MATLAB程序如下所示。

```
clear all
clc
% 定义训练样本矢量
% P 为输入矢量
P = [-1:0.05:1];
% T 为目标矢量
T = sin(2*pi*P)+0.1*randn(size(P));
% 绘制训练样本数据点
figure(1)
plot(P,T,'+');
title('绘制样本数据点');
xlabel('输入变量');
ylabel('目标变量');
hold on;
% 绘制不含噪声的正弦曲线
figure(2)
```

```
plot(P,sin(2*pi*P),':');
title('绘制不含噪声的数据点');
xlabel('输入变量');
ylabel('目标变量');
hold on;
% 定义验证样本
% 验证样本的输入矢量
v.P = [-0.975:0.05:0.975];
% 验证样本的目标矢量
v.T = sin(2*pi*v.P)+0.1*randn(size(v.P));
% 创建一个新的前向神经网络
net=newff(minmax(P),[5,1],{'tansig','purelin'},'traingdx');
% 设置训练参数
net.trainParam.epochs = 500;
net.trainParam.goal = 1e-6;
net = init(net);
% 训练 BP 网络
[net,tr]=train(net,P,T,[],[],v);
% 对 BP 网络进行仿真
A = sim(net,P);
% 计算仿真误差
E = T - A;
MSE=mse(E)
% 绘制仿真拟合结果曲线
figure(3)
plot(P,A,P,T,'+');
title('绘制样本数据点');
```

xlabel('输入变量');

ylabel('目标变量');

神经网络的输出误差变化图如图16-7所示。由图可见，当训练至第114步时，训练提前停止，此时的网络误差为0.011 695。

训练后的仿真数据拟合曲线如图16-8所示，效果是相当满意的。

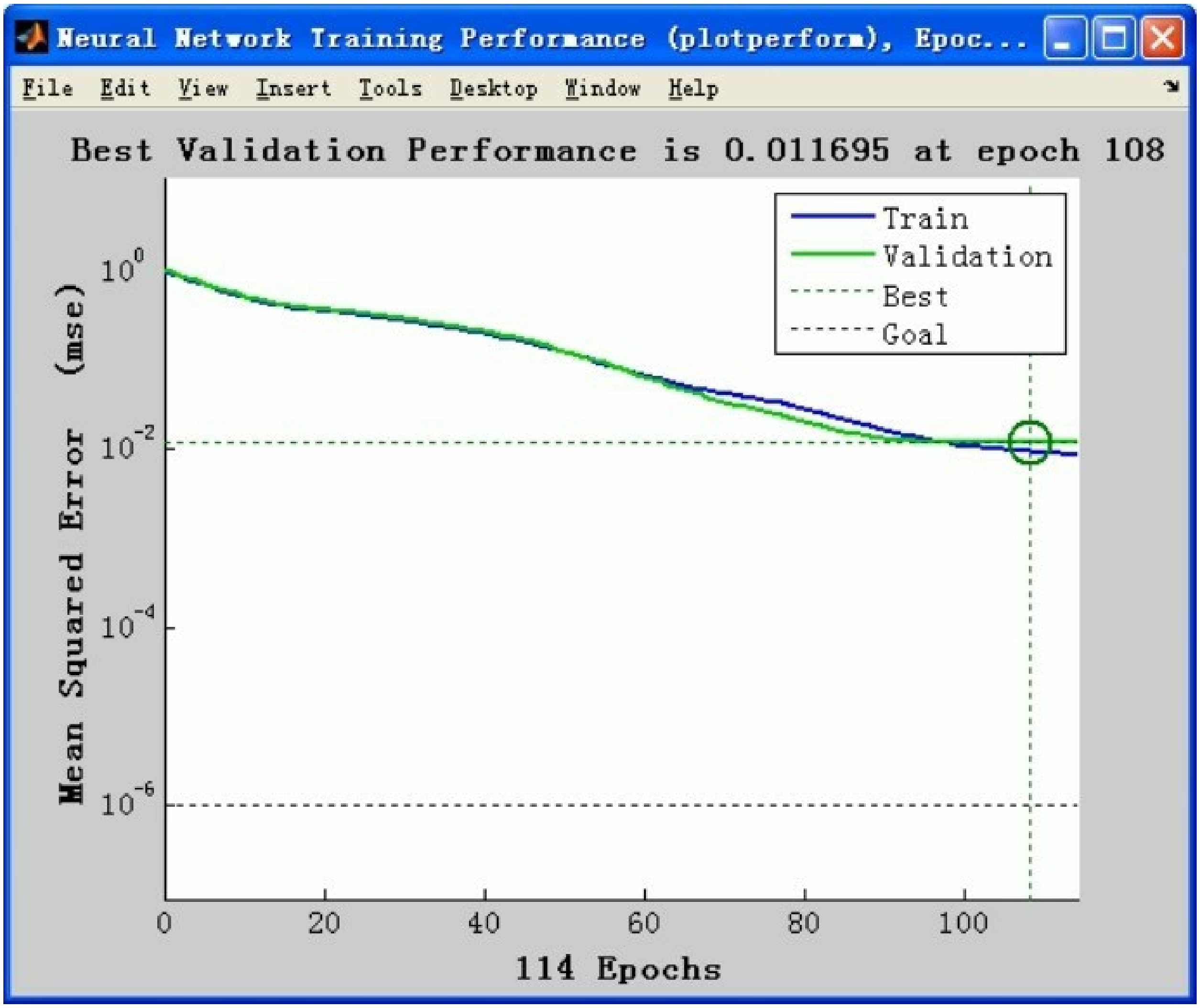


图16-7 神经网络的输出误差变化图

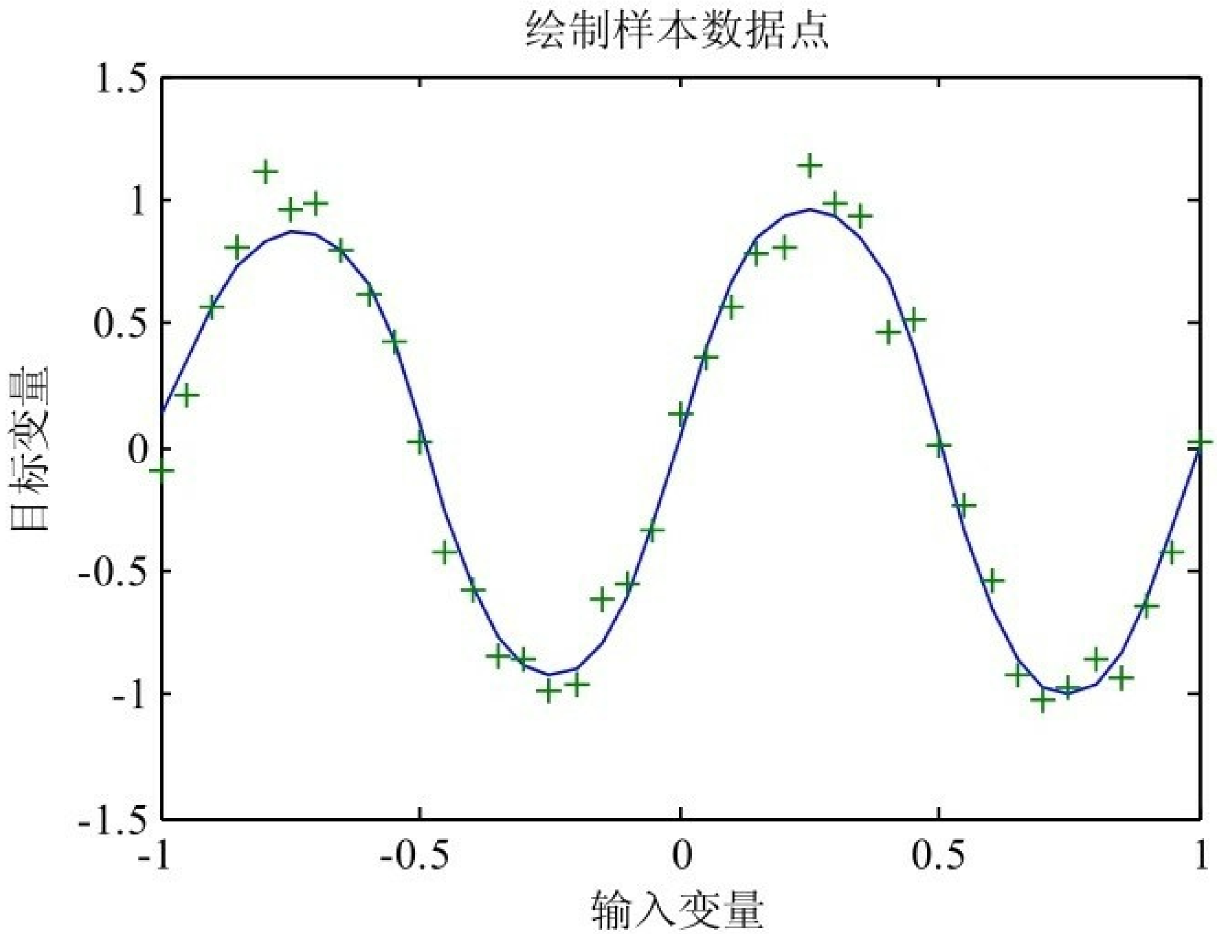


图16-8 训练后的仿真数据拟合曲线

训练得到的神经网络仿真所得均方误差如下。

```
>> MSE
MSE =
   0.0093
```

16.1.2 数据预测

神经网络具有非线性的基本特征，并具有并行结构和学习能力，对应外部激励，能给出相应的输出。

BP_Adaboost模型即把BP神经网络作为弱分类器，反复训练BP神经网络预测样本输出，通过Adaboost算法得到多个BP神经网络弱分类器组

成的强分类器。

【例 16-2】利用 BP_Adaboost 算法弱分类器预测公司产品同比销售状况。其中，负数表示同比销售为负增长，正数表示同比销售为正增长。

解：根据数据维数，采用BP神经网络的结构为10-10-1，共训练生成10 个神经网络弱分类器，最后用10个弱分类器组成强分类器对公司产品同比销售状态进行分离。

基于BP_Adaboost模型的处理步骤如下所示。

（1）数据选择和网络初始化。

（2）弱分类器预测。

（3）计算预测序列权重。

（4）测试数据权重调整。

（5）强分类函数生成。

MATLAB程序如下所示。

```
% 清空环境变量
clc
clear
% 下载数据
load data input_train output_train input_test output_test
% 权重初始化
[mm,nn]=size(input_train);
D(1,:)=ones(1,nn)/nn;
% 弱分类器分类
K=10;
for i=1:K
    %训练样本归一化
    [inputn,inputps]=mapminmax(input_train);
```

```
[outputn,outputps]=mapminmax(output_train);
error(i)=0;
%BP神经网络构建
net=newff(inputn,outputn,8);
net.trainParam.epochs=8;
net.trainParam.lr=0.04;
net.trainParam.goal=0.00005;
%BP神经网络训练
net=train(net,inputn,outputn);
%训练数据预测
an1=sim(net,inputn);
test_simu1(i,:)=mapminmax('reverse',an1,outputps);
%测试数据预测
inputn_test =mapminmax('apply',input_test,inputps);
an=sim(net,inputn_test);
test_simu(i,:)=mapminmax('reverse',an,outputps);
%统计输出效果
kk1=find(test_simu1(i,:)>0);
kk2=find(test_simu1(i,:)<0);
aa(kk1)=1;
aa(kk2)=-1;
%统计错误样本数
for j=1:nn
    if aa(j)～=output_train(j);
        error(i)=error(i)+D(i,j);
    end
end
```

```
    %弱分类器i权重
    at(i)=0.5*log((1-error(i))/error(i));
    %更新D值
    for j=1:nn
        D(i+1,j)=D(i,j)*exp(-at(i)*aa(j)*test_simu1(i,j));
    end
    %D值归一化
    Dsum=sum(D(i+1,:));
    D(i+1,:)=D(i+1,:)/Dsum;
end
% 强分类器分类结果
output=sign(at*test_simu);
% 分类结果统计,统计强分类器每类分类错误个数
kkk1=0;
kkk2=0;
for j=1:350
    if output(j)==1
        if output(j)～=output_test(j)
            kkk1=kkk1+1;
        end
    end
    if output(j)==-1
        if output(j)～=output_test(j)
            kkk2=kkk2+1;
        end
    end
end
```

```
kkk1
kkk2
disp('第一类分类错误 第二类分类错误 总错误');
% 窗口显示
disp([kkk1 kkk2 kkk1+kkk2]);
plot(output)
hold on
plot(output_test,'g')
%统计弱分离器效果
for i=1:K
    error1(i)=0;
    kk1=find(test_simu(i,:)>0);
    kk2=find(test_simu(i,:)<0);
    aa(kk1)=1;
    aa(kk2)=-1;
    for j=1:350
        if aa(j)~=output_test(j);
            error1(i)=error1(i)+1;
        end
    end
end
disp('统计弱分类器分类效果');
error1
disp('强分类器分类误差率')
(kkk1+kkk2)/350
disp('弱分类器分类误差率')
(sum(error1)/(K*350))
```

BP神经网络训练的示意图如图16-9所示。

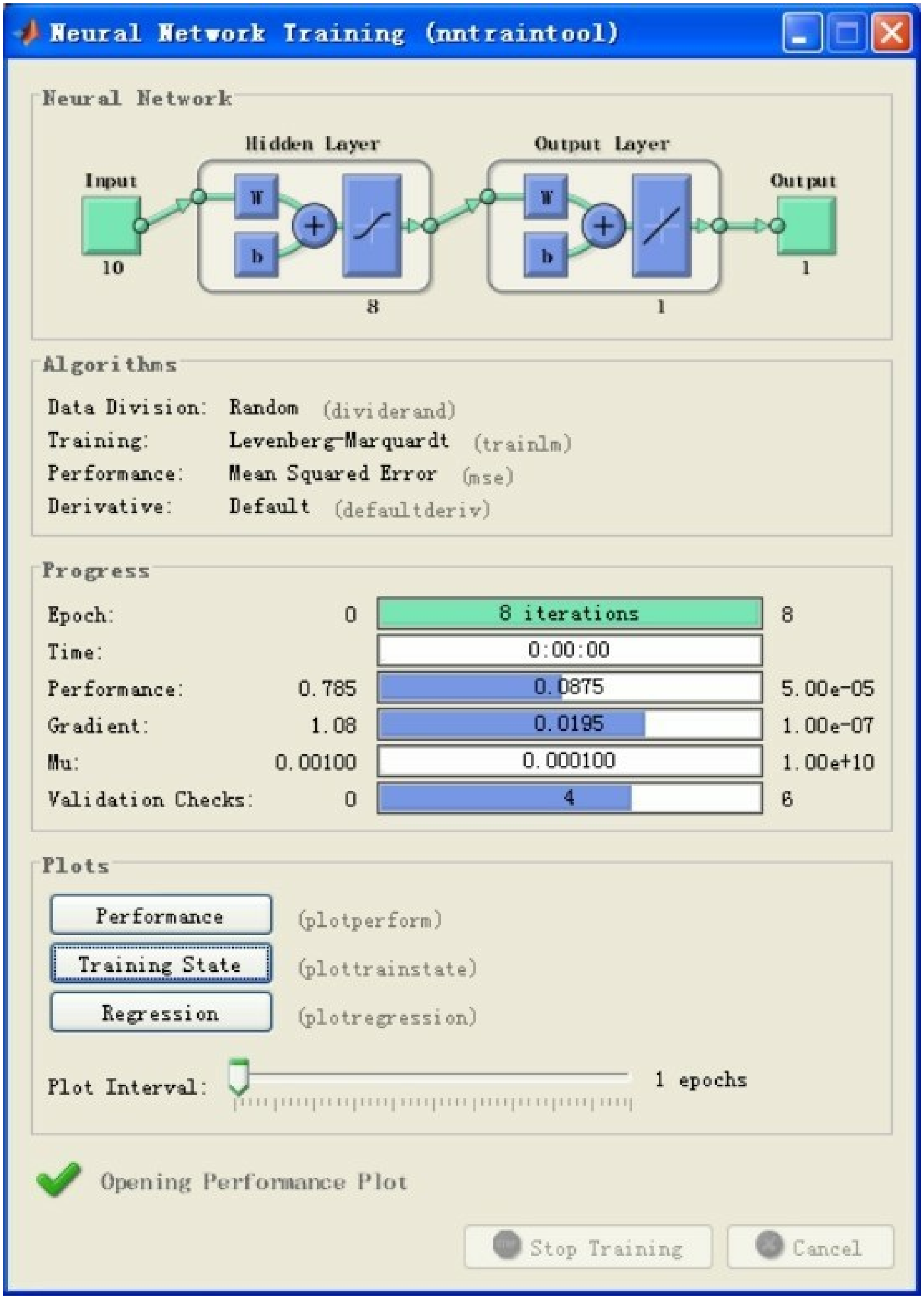


图16-9 BP神经网络训练的示意图

MATLAB代码运行得到如下所示结果。

```
kkk1 =
    0
kkk2 =
    0
第一类分类错误 第二类分类错误 总错误
    0    0    0
统计弱分类器分类效果
error1 =
    350    350    350    350    350    350    350    350    350    350
强分类器分类误差率
ans =
    0
弱分类器分类误差率
ans =
    1
```

公司同比销售样本数据共有350组，采用10个BP弱分类器组成的强分类器。其中，弱分类器分类误差率为1，强分类器分类误差为0，说明BP_Adaboost算法达到预期效果。

BP_Adaboost 除了可以设计强分类器之外，还可以预测强预测器。采用 BP_Adaboost预测强预测器的MATLAB程序如下所示。

```
%clear all
%clc
%下载数据
load data1 input output
% 权重初始化
k=rand(1,2000);
```

```
[m,n]=sort(k);
%训练样本
input_train=input(n(1:1900),:)';
output_train=output(n(1:1900),:)';
%测试样本
input_test=input(n(1901:2000),:)';
output_test=output(n(1901:2000),:)';
%样本权重
[mm,nn]=size(input_train);
D(1,:)=ones(1,nn)/nn;
%训练样本归一化
[inputn,inputps]=mapminmax(input_train);
[outputn,outputps]=mapminmax(output_train);
K=10;
for i=1:K
    %弱预测器训练
    net=newff(inputn,outputn,5);
    net.trainParam.epochs=50;
    net.trainParam.lr=0.05;
    net=train(net,inputn,outputn);
    %弱预测器预测
    an1=sim(net,inputn);
    BPoutput=mapminmax('reverse',an1,outputps);
    %预测误差
    erroryc(i,:)=output_train-BPoutput;
    %测试数据预测
    inputn1=mapminmax('apply',input_test,inputps);
```

```
    an2=sim(net,inputn1);
    test_simu(i,:)=mapminmax('reverse',an2,outputps);
    %调整D值
    Error(i)=0;
    for j=1:nn
        if abs(erroryc(i,j))>0.3 %较大误差
            Error(i)=Error(i)+D(i,j);
            D(i+1,j)=D(i,j)*1.1;
        else
            D(i+1,j)=D(i,j);
        end
    end
    %计算弱预测器权重
    at(i)=0.5/exp(abs(Error(i)));
    %D值归一化
    D(i+1,:)=D(i+1,:)/sum(D(i+1,:));
end
% 强预测器预测
at=at/sum(at);
% 结果统计,强分离器效果
output=at*test_simu;
error=output_test-output;
plot(abs(error))
hold on
for i=1:8
error1(i,:)=test_simu(i,:)-output;
end
```

```
plot(mean(abs(error1)))
title('强预测器预测误差绝对值','fontsize',10)
xlabel('预测样本','fontsize',10)
ylabel('误差绝对值','fontsize',10)
legend('强预测器预测','弱预测器预测')
```

神经网络训练的误差曲线如图16-10所示。

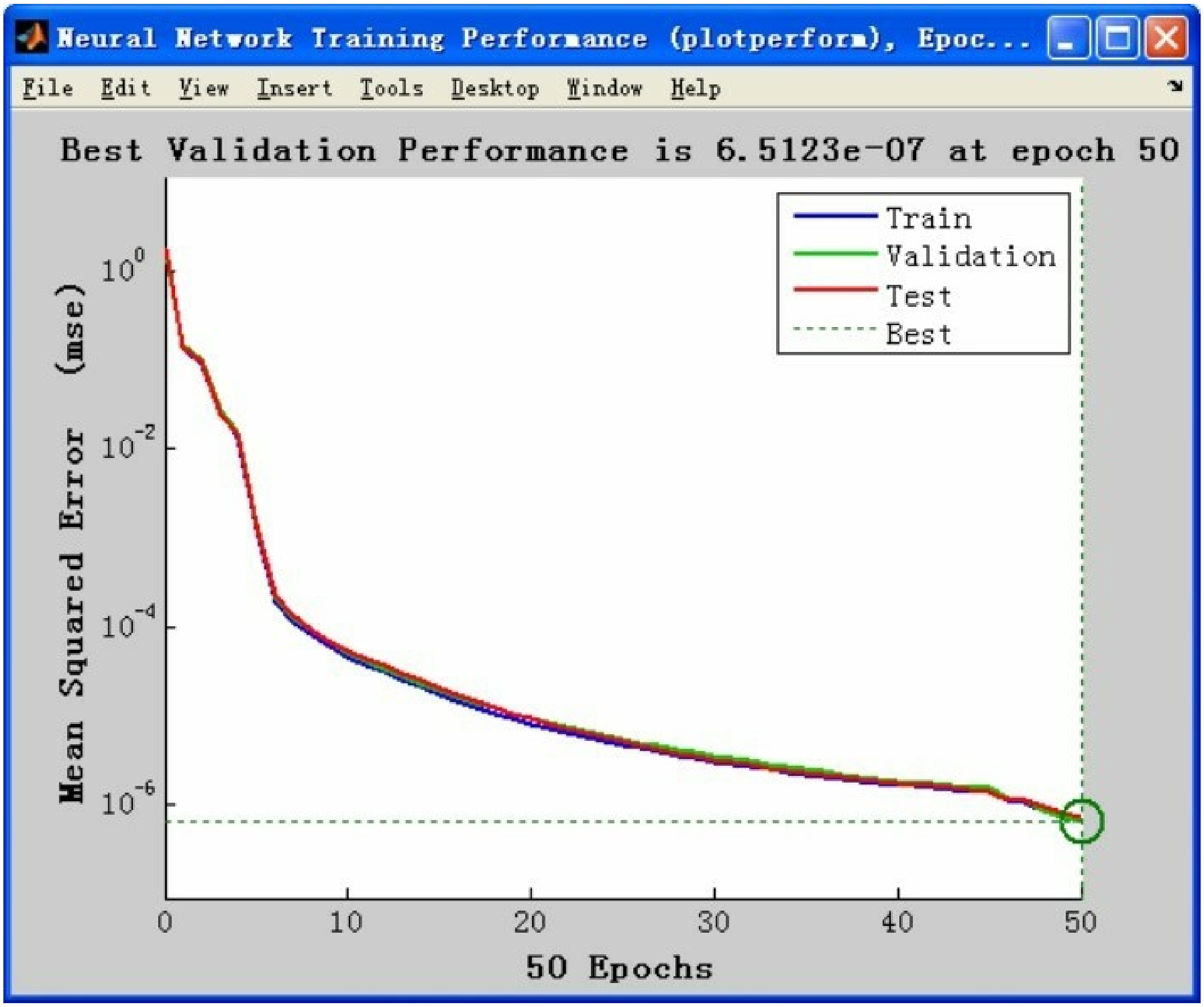


图16-10 神经网络训练的误差曲线

在上述代码中，共有350组预测样品，弱预测器共有10个BP神经网络构成弱预测序列，弱、强预测器预测的平均误差绝对值如图16-11所示。

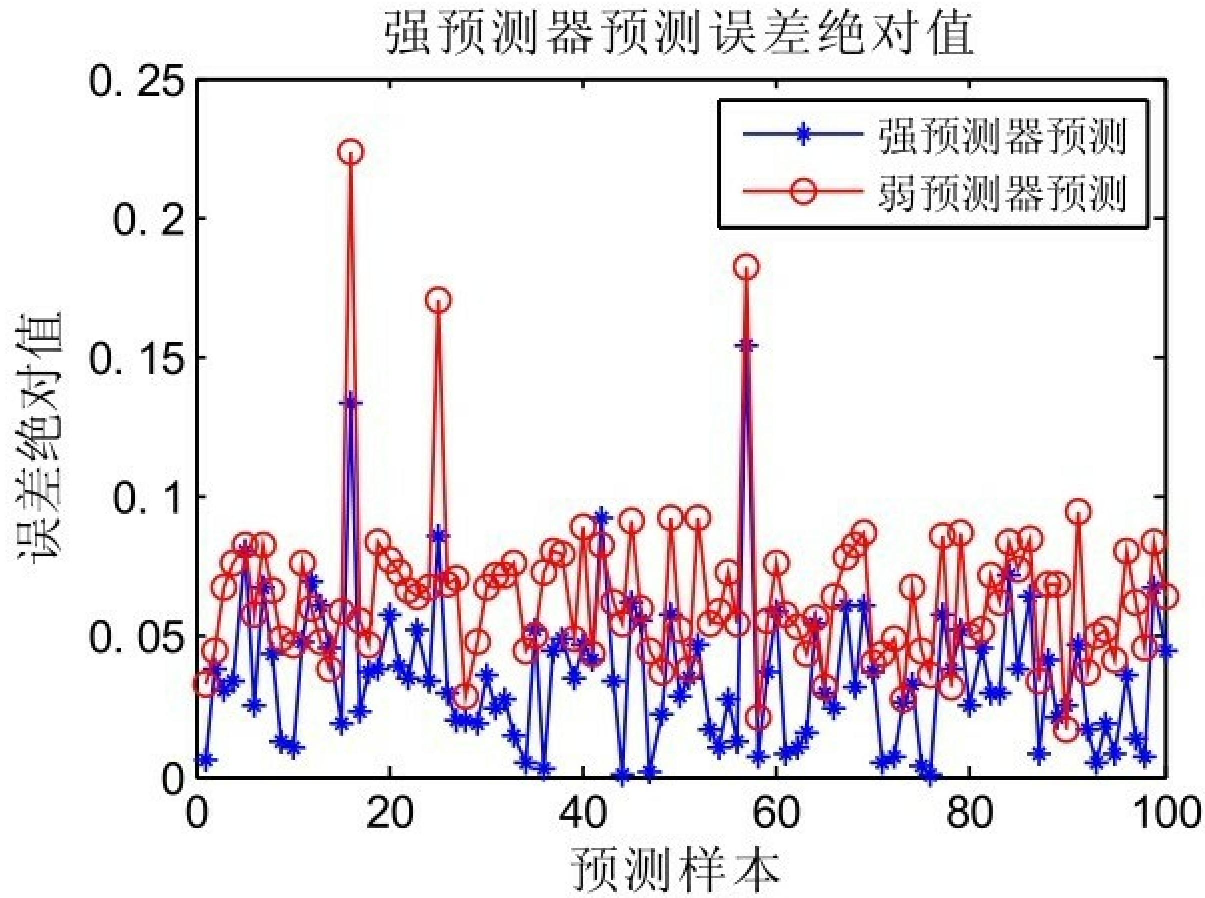


图16-11 弱、强预测器预测的平均误差绝对值

16.1.3 函数逼近

BP网络有很强的映射能力，主要用于模式识别分类、函数逼近、函数压缩等。下面将通过实例来说明BP网络在函数逼近方面的应用。

【例16-3】要求设计一个BP 网络，逼近函数g(x)=2-sin(k·pi/3·x)，实现对该非线性函数的逼近。其中，分别令k=1，3，6进行仿真，通过调节参数（如隐藏层节点个数等）得出信号的频率与隐含层节点之间，隐含层节点与函数逼近能力之间的关系。

解：假设频率参数 k=1，绘制要逼近的非线性函数的曲线。函数的曲线如图 16-12所示。

k=1;

```
k=1;
p=[-1:.05:8];
t=1+sin(k*pi/4*p);
plot(p,t,'-');
title('要逼近的非线性函数');
xlabel('时间');
ylabel('非线性函数');
```

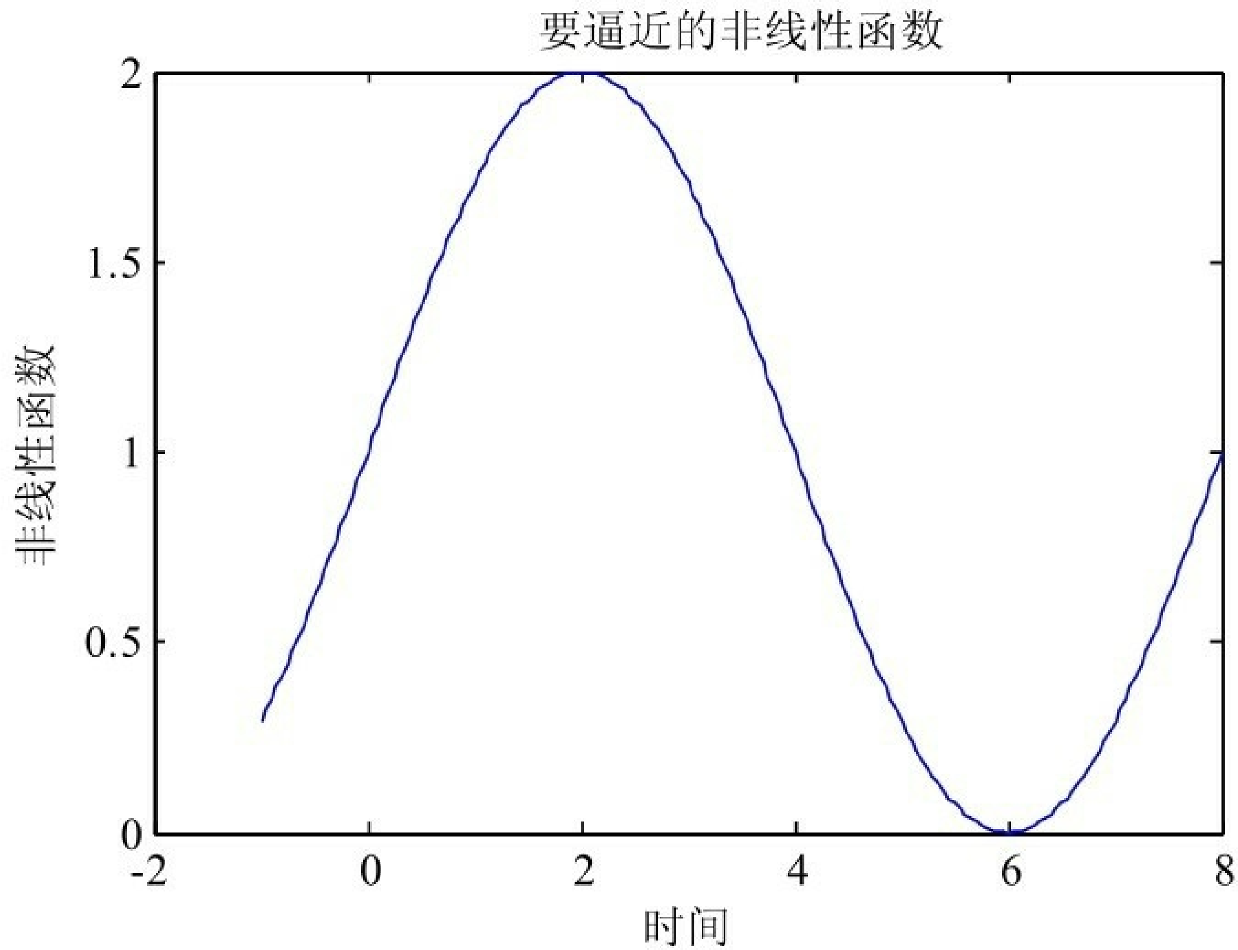


图16-12 要逼近的非线性函数曲线

应用newff()函数建立BP网络结构。隐含层神经元数目n可以改变，暂设为n=5，输出层有一个神经元。选择隐含层和输出层神经元传递函数分别为tansig函数和purelin函数，网络训练的算法采用Levenberg–

Marquardt算法trainlm。

```
n=5;
net = newff(minmax(p),[n,1],{'tansig' 'purelin'},'trainlm');
```

对于初始网络，可以应用sim（）函数观察网络输出。

```
y1=sim(net,p);
figure;
plot(p,t,'-',p,y1,':')
title('未训练网络的输出结果');
xlabel('时间');
ylabel('仿真输出--原函数-');
```

同时绘制网络输出曲线，并与原函数相比较，结果如图16-13所示。

其中，实线代表要逼近的非线性函数曲线；虚线代表未经训练的函数曲线。

因为使用newff( )函数建立函数网络时，权值和阈值的初始化是随机的，所以网络输出结构很差，根本达不到函数逼近的目的，每次运行的结果也有时不同。

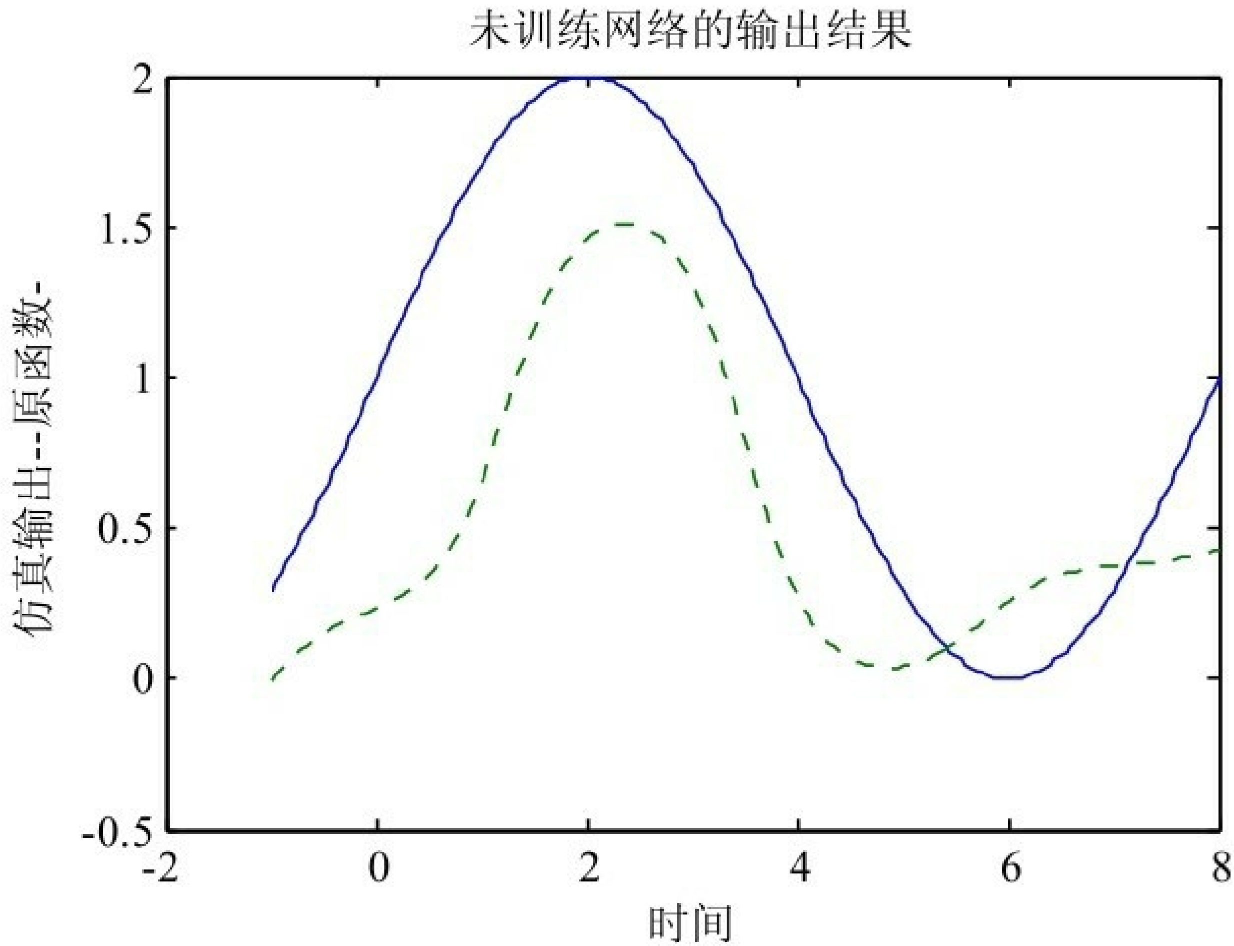


图16-13 未训练网络的输出结果

应用train（）函数对网络进行训练之前，需要预先设置网络训练参数。将训练时间设置为 40，训练精度设置为 0.005，其余参数使用缺省值。训练后得到的误差变化过程如图16-14所示。

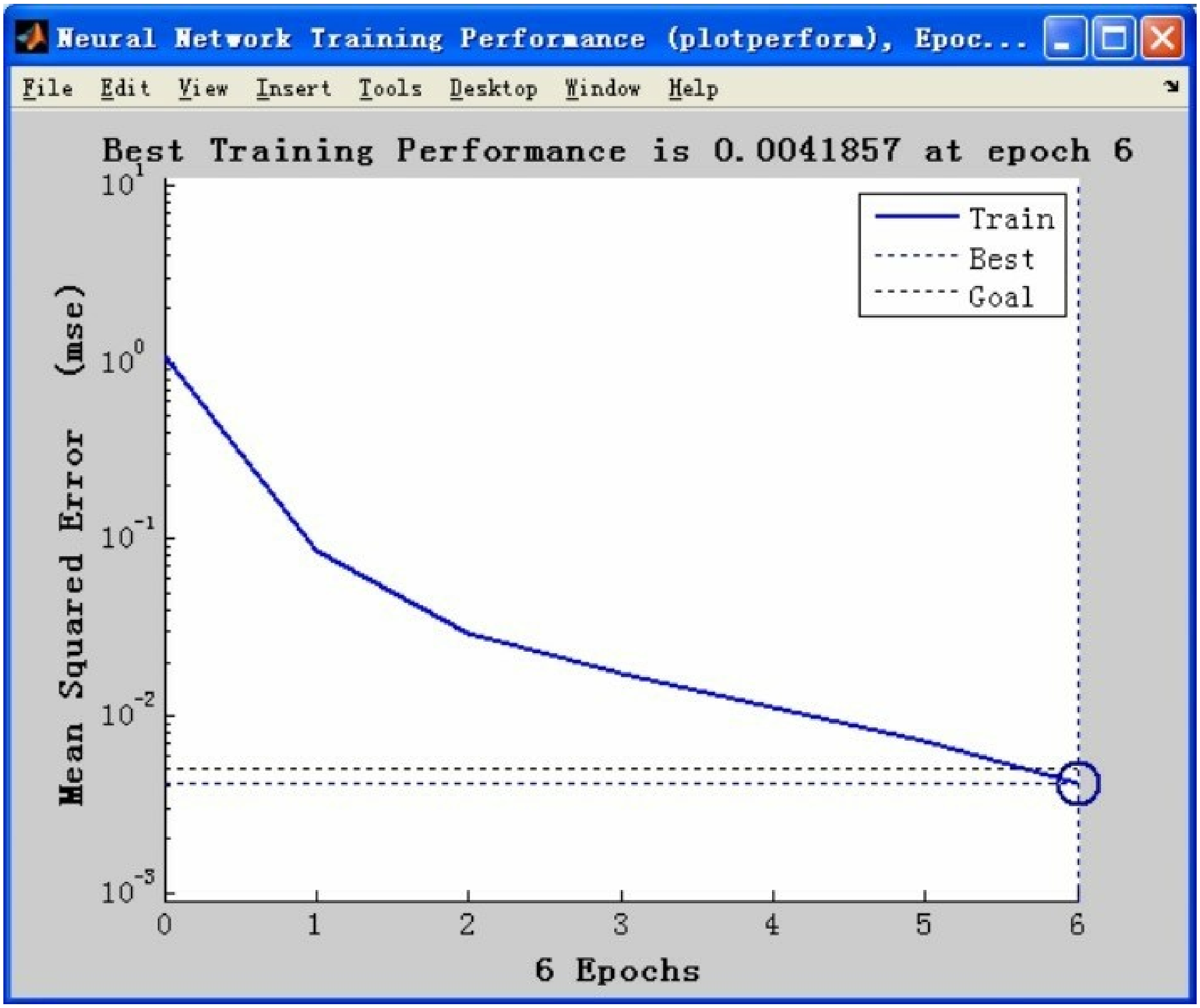


图16-14 训练过程

```
%网络训练时间设置为40
net.trainParam.epochs=40;
%网络训练精度设置为0.005
net.trainParam.goal=0.005;
%开始训练网络
net=train(net,p,t);
```

从以上结果可以看出，网络训练速度很快。

对于训练好的网络进行仿真。

```
y2=sim(net,p);
```

```
figure;
plot(p,t,'-',p,y1,':',p,y2, '--')
title('训练后网络的输出结果');
xlabel('时间');
ylabel('仿真输出');
```

绘制网络输出曲线，并与原始非线性函数曲线以及未训练网络的输出结果曲线相比较，比较出来的结果如图16-15所示。

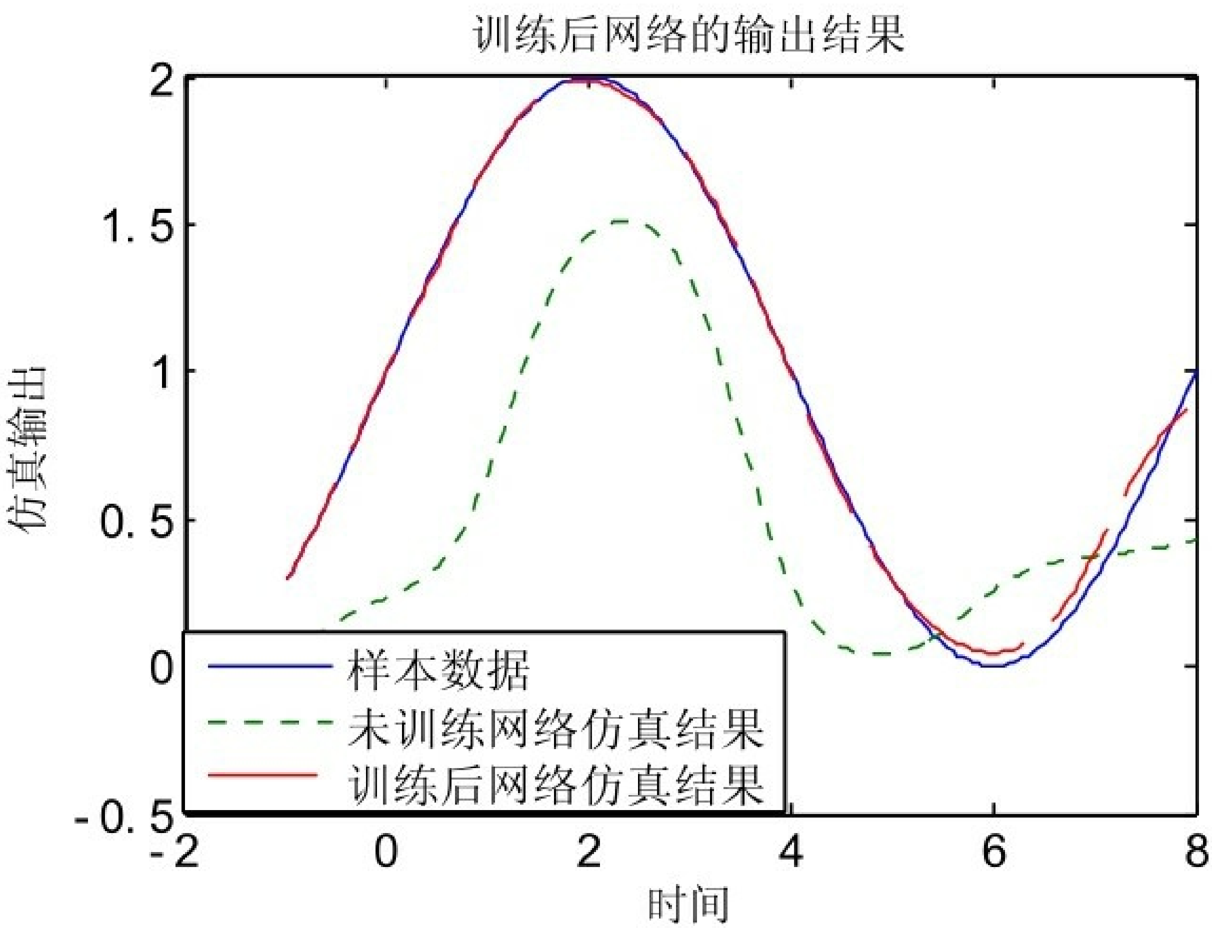


图16-15 训练后网络的输出结果

其中，“——”代表要逼近的非线性函数曲线；“ ----”代表未经训练的函数曲线；“ ——”代表经过训练的函数曲线；

从图 16-15 中可以看出，得到的曲线和原始的非线性函数曲线很接近。这说明经过训练后，BP网络对非线性函数的逼近效果比较好。

以上是当k=1时候的逼近效果，下面再看不同频率下的逼近效果

改变非线性函数的频率和 BP 函数隐含层神经元的数目，对于函数逼近的效果有一定的影响。网络非线性程度越高，对于 BP 网络的要求越高，则相同的网络逼近效果要差一些；隐含层神经元的数目对于网络逼近效果也有一定影响，一般来说隐含层神经元数目越多，则BP网络逼近非线性函数的能力越强。

下面通过改变频率参数和非线性函数的隐含层神经元数目来加以比较证明。

（1）频率参数设为k=3时，当隐含层神经元数目分别取n=5、n=10时，得到了训练后的网络输出结果如图16-16和图16-17所示。

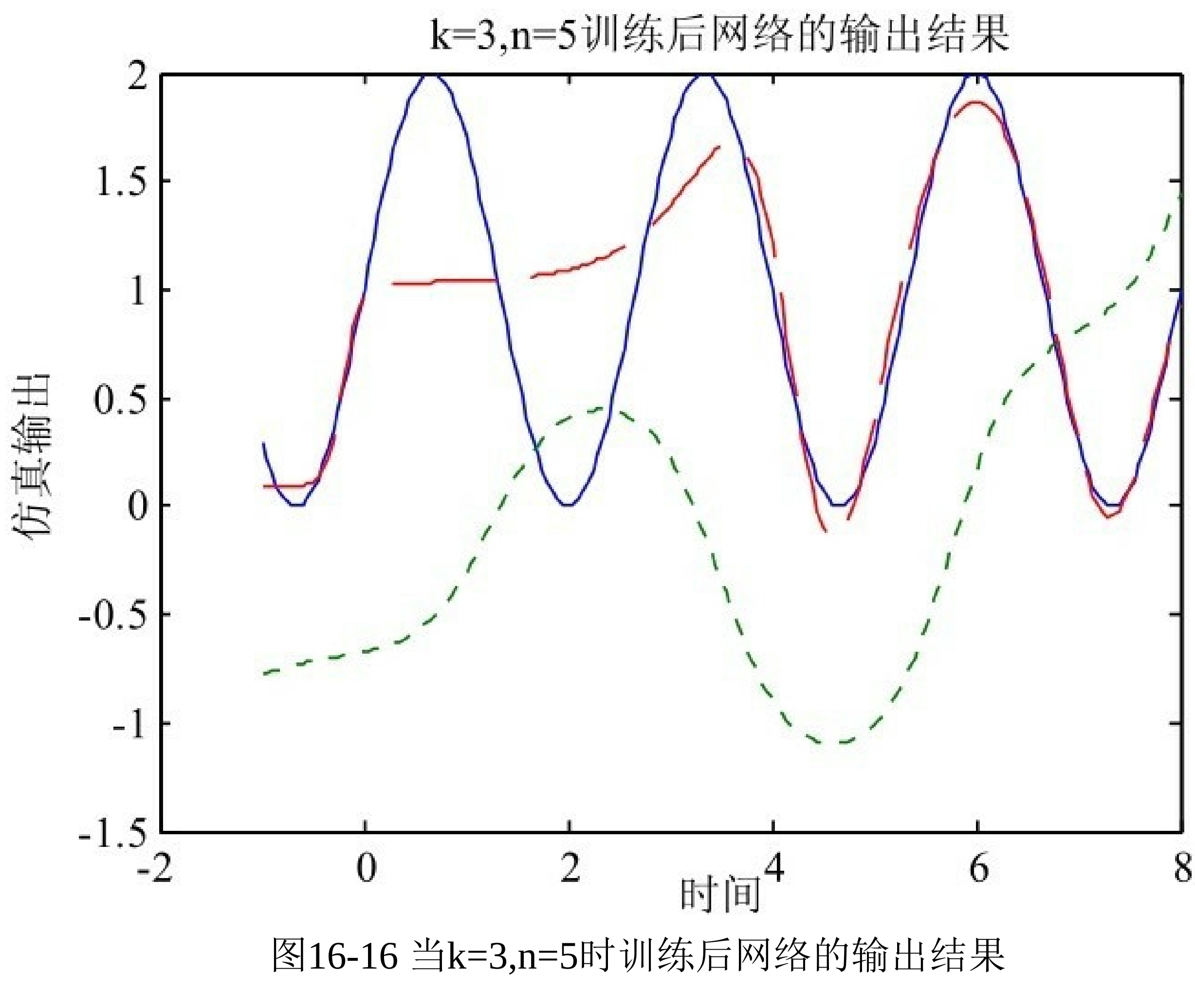


图16-16 当k=3,n=5时训练后网络的输出结果

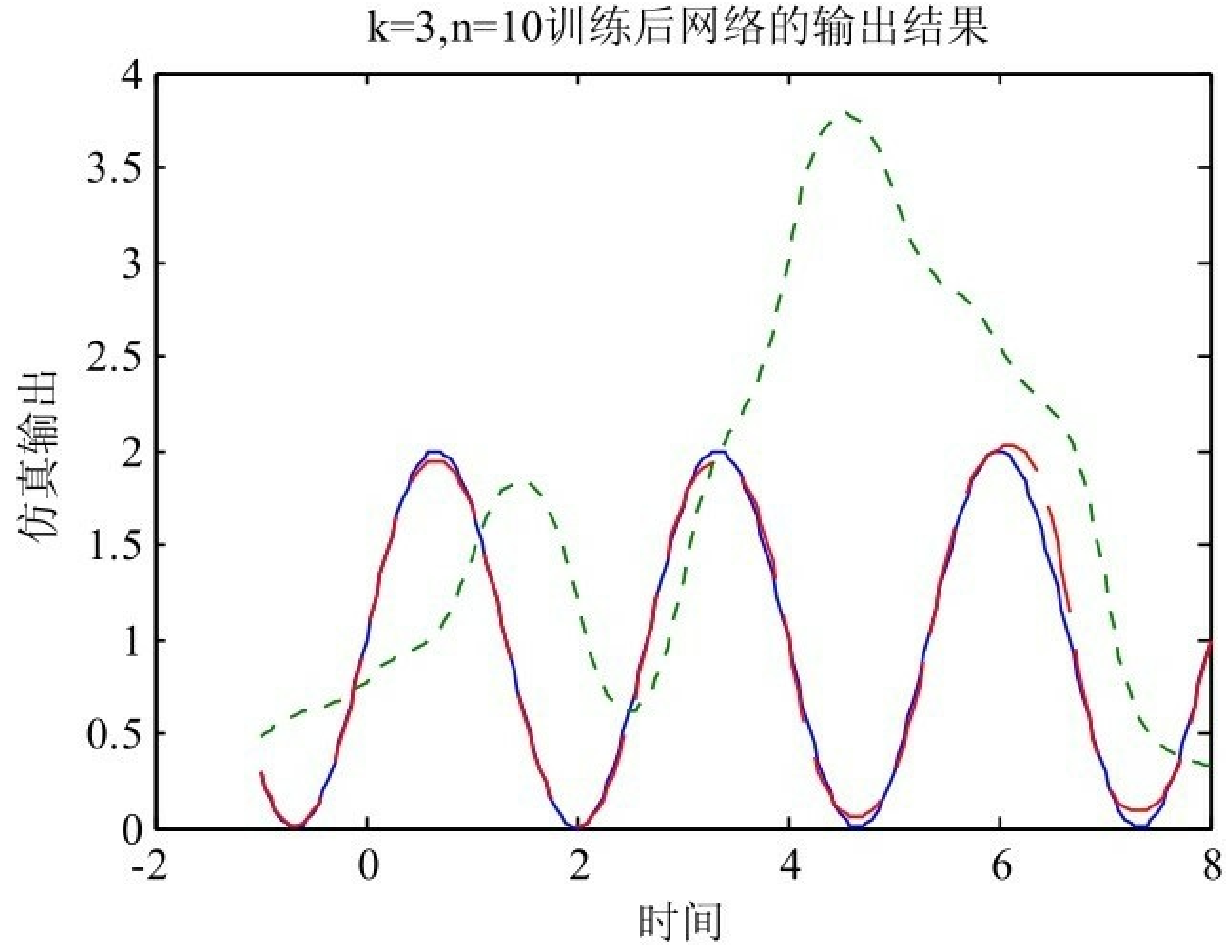


图16-17 当k=3,n=10时训练后网络的输出结果

其中，“——”代表要逼近的非线性函数曲线；“----　”代表未经训练的函数曲线；“—— ”代表经过训练的函数曲线。

（2）频率参数设为k=6，当隐含层神经元数目分别取n=5、n=10时，得到了训练后的网络输出结果如图16-18和图16-19所示。

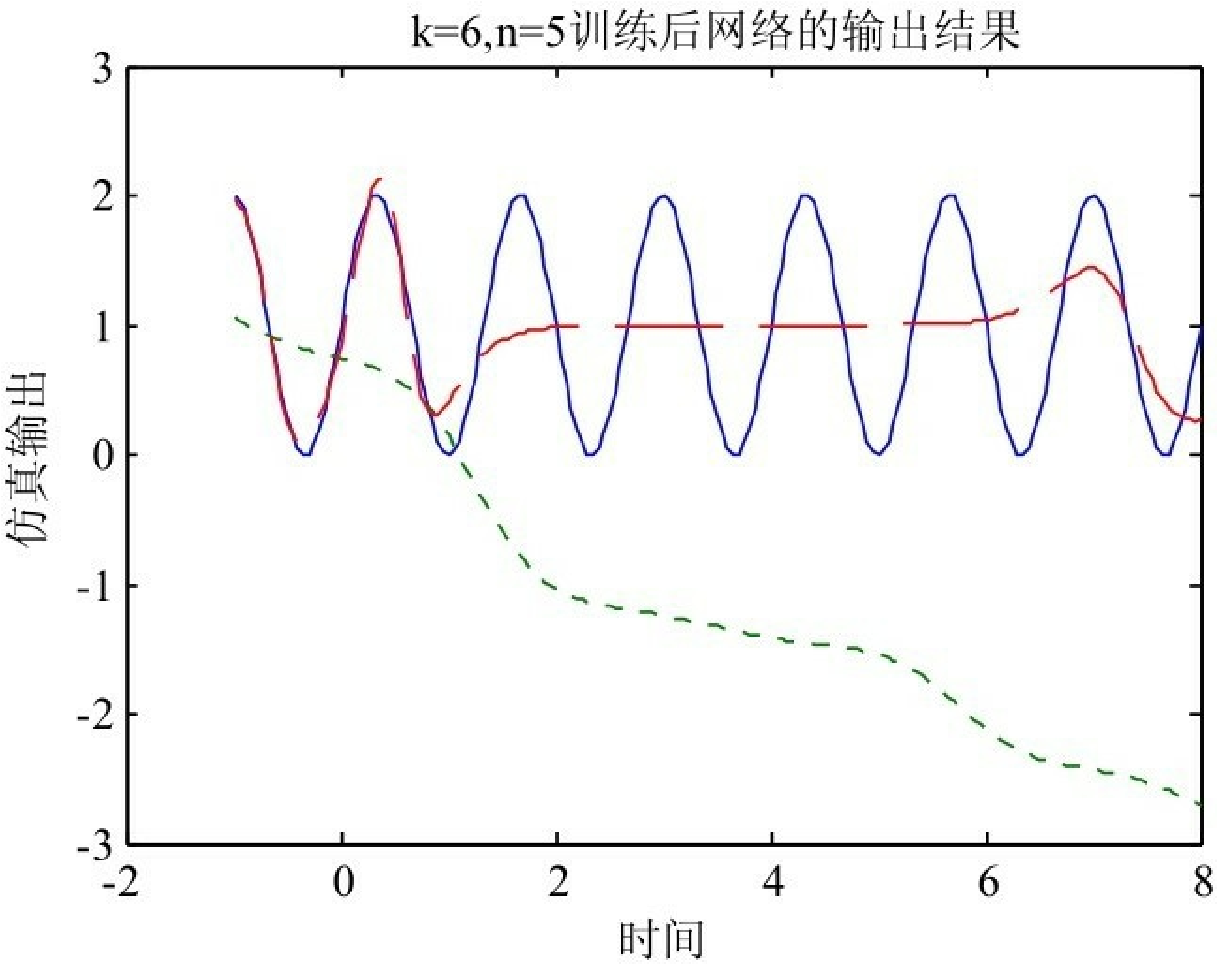


图16-18 当k=6，n=5时训练后网络的输出结果

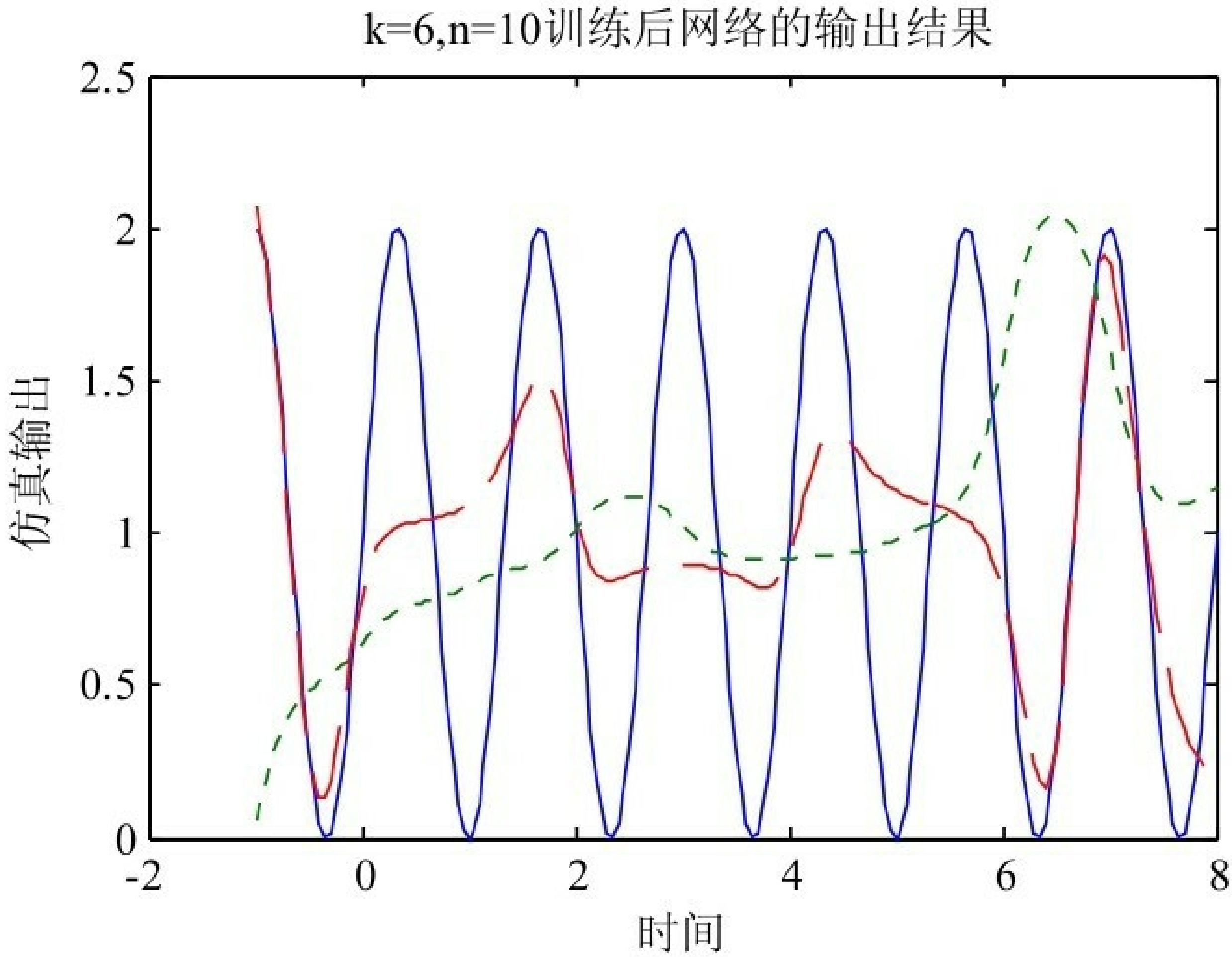


图16-19 当k=6，n=10时训练后网络的输出结果

其中：“——”代表要逼近的非线性函数曲线；“----　”代表未经训练的函数曲线；“—— ”代表经过训练的函数曲线。

通过上述仿真结果可知，当k=1，n=5时；k=3，n=3时；k=6，n=10时，BP神经网络分别对函数取得了较好的逼近效果。由此可见，n 取不同的值对函数逼近的效果有很大的影响。改变BP网络隐含层神经元的数目，可以改变BP神经网络对于函数的逼近效果。隐含层神经元数目越多，则BP网络逼近非线性函数的能力越强。

16.2 PID神经网络控制

鉴于被控对象的复杂性、大规模和确定性、分布性，要实现自动控制，基于传统精确数学模型的控制理论就显现出极大的局限性。

智能控制的概念和原理主要是针对被控对象、环境、控制目标或任务的复杂性而提出来的。

PID 控制器是一个在工业控制应用中常见的反馈回路部件。这个控制器把收集到的数据和一个参考值进行比较，然后把这个差别用于计算新的输入值，这个新的输入值的目的是可以让系统的数据达到或者保持在参考值。

PID在本质上是线性控制规律，具有传统控制理论的弱点——只适合于线性SISO系统，在复杂系统中控制效果不佳。

PIDNN（PID神经元网络）的基础是分别定义了具有比例、积分、微分功能的神经元，从而将PID控制规律融合进神经元网络之中。PIDNN的各层神经元个数、连接方式、连接权重初值是按PID控制规律的基本原则来确定的。

PIDNN的主要特点如下。

·PIDNN 属于交层前向神经元网络。

·PIDNN 参照PID 的控制规律的要求构成，结构比较简单、规范。

·PIDNN 的初值按 PID 控制规律的基本原则确定，加快了收敛速度，不易陷入极小点；更重要的是，可以利用现有的PID控制的大量经验数据确定网络权重初值，使控制系统保持初始稳定，使系统的全局稳定成为可能。

·PIDNN 可采用“无教师”的学习方式，根据控制效果进行在线自学习和调整，使系统具备较好的性能。

·PIDNN 可同时适用于SISO 以及MIMO控制系统。

PIDNN的结构形式分为SPIDNN(single-out PIDNN)和MPIDNN(Multi-output PIDNN)，结构示意图分别如图16-20和图16-21所示。

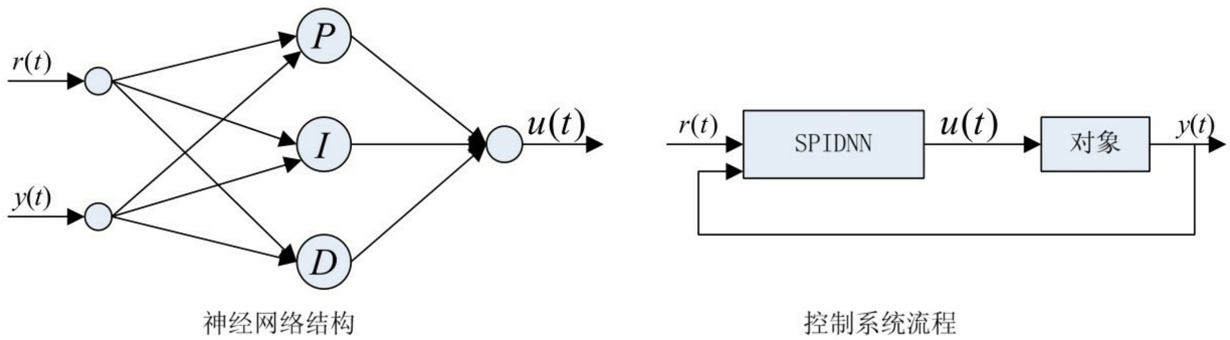


图16-20 SPIDNN结构示意图

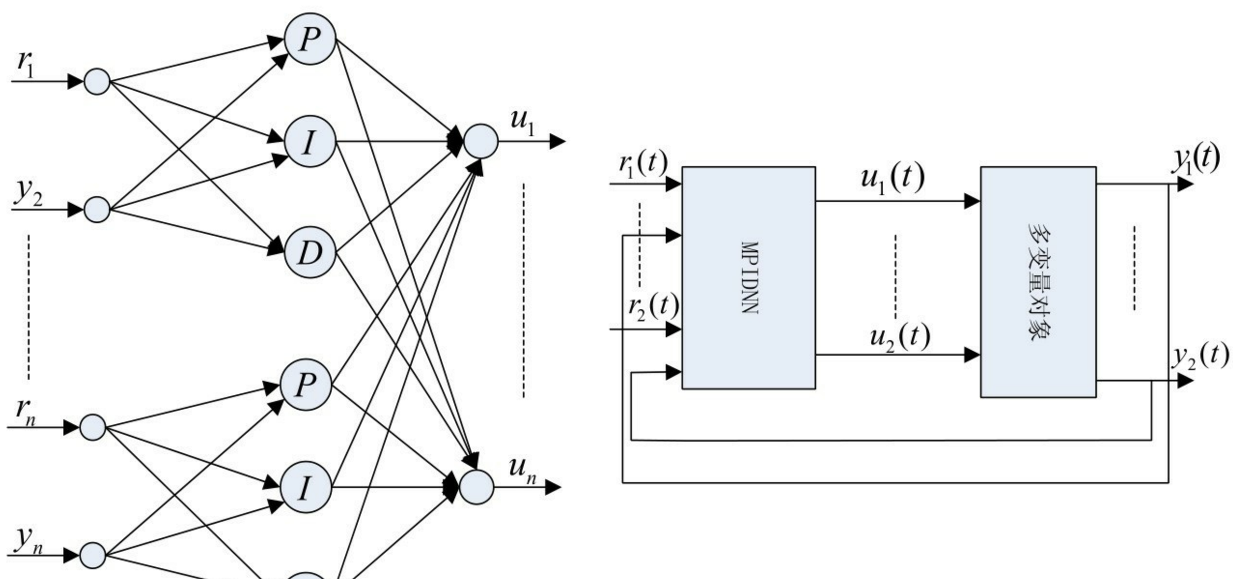


图16-21 MPIDNN结构示意图

目前应用较多的神经元模型，一般只考虑了神经元的静态特性，只把神经元看作是一个具有静态输入——输出影射关系的单元，只能处理静态信息。

对于动态信息的处理，一般是通过神经网络的互联方式的动态结构

进行，PID 神经元中，不仅有具备静态非线性影射功能的比例元，还有可处理动态信息的积分和微分元。

PID 神经元既具有一般神经元的共性，又具备着不同的特性，尤其是积分元和微分元的引入，使作为数字基石的微积分概念同神经元网络的基本单位融合为一体，增强了神经元处理信息的能力，充实和完善了神经元的种类和内涵，使神经元网络更加丰富多样。

将PID和一般神经元网络融合起来的方法包括两个步骤。

① 将PID功能引入神经网络的神经元中，构成PID 神经元。

② 按照PID神经元的控制规律的基本模式，用这些基本神经元构成新的神经元网络，并找到合理有效的计算与学习方法。

【例16-4】根据PID神经元网络控制器原理，在MATLAB中编程实现PID神经元网络3个控制变量的控制系统。

解：PID神经网络控制器的MATLAB代码如下所示。

```
clc
clear
%网络结构初始化
rate1=0.005;
rate2=0.002;
rate3=0.0002;
k=0.4;K=3;
y_1=zeros(3,1);y_2=y_1;y_3=y_2;    %输出值
u_1=zeros(3,1);u_2=u_1;u_3=u_2;    %控制率
h1i=zeros(3,1);h1i_1=h1i;      %第一个控制量
h2i=zeros(3,1);h2i_1=h2i;      %第二个控制量
h3i=zeros(3,1);h3i_1=h3i;      %第三个空置量
x1i=zeros(3,1);x2i=x1i;x3i=x2i;x1i_1=x1i;x2i_1=x2i;x3i_1=x3i;    %隐含层输出
```

```
%权值初始化
k0=0.03;
%第一层权值
w11=k0*rand(3,2);w11_1=w11;w11_2=w11_1;
w12=k0*rand(3,2);w12_1=w12;w12_2=w12_1;
w13=k0*rand(3,2);w13_1=w13;w13_2=w13_1;
%第二层权值
w21=k0*rand(1,9);w21_1=w21;w21_2=w21_1;
w22=k0*rand(1,9);w22_1=w22;w22_2=w22_1;
w23=k0*rand(1,9);w23_1=w23;w23_2=w23_1;
%值限定
ynmax=1;ynmin=-1; %系统输出值限定
xpmax=1;xpmin=-1; %P节点输出限定
qimax=1;qimin=-1; %I节点输出限定
qdmax=1;qdmin=-1; %D节点输出限定
uhmax=1;uhmin=-1; %输出结果限定
%% 网络迭代优化
for k=1:1:200
    % 控制量输出计算
    %--------------------------------网络前向计算--------------------------
    %系统输出
    y1(k)=
(0.4*y_1(1)+u_1(1)/(1+u_1(1)^2)+0.2*u_1(1)^3+0.5*u_1(2))+0.3*y_1(2);
    y2(k)=
(0.2*y_1(2)+u_1(2)/(1+u_1(2)^2)+0.4*u_1(2)^3+0.2*u_1(1))+0.3*y_1(3);
    y3(k)=
(0.3*y_1(3)+u_1(3)/(1+u_1(3)^2)+0.4*u_1(3)^3+0.4*u_1(2))+0.3*y_1(1);
```

```
r1(k)=0.7;r2(k)=0.4;r3(k)=0.6; %控制目标
%系统输出限制
yn=[y1(k),y2(k),y3(k)];
yn(find(yn>ynmax))=ynmax;
yn(find(yn<ynmin))=ynmin;
%输入层输出
x1o=[r1(k);yn(1)];x2o=[r2(k);yn(2)];x3o=[r3(k);yn(3)];
%隐含层
x1i=w11*x1o;
x2i=w12*x2o;
x3i=w13*x3o;
%比例神经元P计算
xp=[x1i(1),x2i(1),x3i(1)];
xp(find(xp>xpmax))=xpmax;
xp(find(xp<xpmin))=xpmin;
qp=xp;
h1i(1)=qp(1);h2i(1)=qp(2);h3i(1)=qp(3);
%积分神经元I计算
xi=[x1i(2),x2i(2),x3i(2)];
qi=[0,0,0];qi_1=[h1i(2),h2i(2),h3i(2)];
qi=qi_1+xi;
qi(find(qi>qimax))=qimax;
qi(find(qi<qimin))=qimin;
h1i(2)=qi(1);h2i(2)=qi(2);h3i(2)=qi(3);
%微分神经元D计算
xd=[x1i(3),x2i(3),x3i(3)];
qd=[0 0 0];
```

```
xd_1=[x1i_1(3),x2i_1(3),x3i_1(3)];
qd=xd-xd_1;
qd(find(qd>qdmax))=qdmax;
qd(find(qd<qdmin))=qdmin;
h1i(3)=qd(1);h2i(3)=qd(2);h3i(3)=qd(3);
%输出层计算
wo=[w21;w22;w23];
qo=[h1i',h2i',h3i'];qo=qo';
uh=wo*qo;
uh(find(uh>uhmax))=uhmax;
uh(find(uh<uhmin))=uhmin;
u1(k)=uh(1);u2(k)=uh(2);u3(k)=uh(3);
%-------------------------------------网络反馈修正---------------------
%计算误差
error=[r1(k)-y1(k);r2(k)-y2(k);r3(k)-y3(k)];
error1(k)=error(1);error2(k)=error(2);error3(k)=error(3);
J(k)=0.5*(error(1)^2+error(2)^2+error(3)^2); %调整大小
ypc=[y1(k)-y_1(1);y2(k)-y_1(2);y3(k)-y_1(3)];
uhc=[u_1(1)-u_2(1);u_1(2)-u_2(2);u_1(3)-u_2(3)];
%隐含层和输出层权值调整
%调整w21
Sig1=sign(ypc./(uhc(1)+0.00001));
dw21=sum(error.*Sig1)*qo';
w21=w21+rate2*dw21+rate3*(w21_1-w21_2);
%调整w22
Sig2=sign(ypc./(uh(2)+0.00001));
dw22=sum(error.*Sig2)*qo';
```

```
w22=w22+rate2*dw22+rate3*(w22_1-w21_2);
%调整w23
Sig3=sign(ypc./(uh(3)+0.00001));
dw23=sum(error.*Sig3)*qo';
w23=w23+rate2*dw23+rate3*(w23_1-w23_2);
%输入层和隐含层权值调整
delta2=zeros(3,3);
wshi=[w21;w22;w23];
for t=1:1:3
    delta2(1:3,t)=error(1:3).*sign(ypc(1:3)./(uhc(t)+0.00000001));
end
for j=1:1:3
    sgn(j)=sign((h1i(j)-h1i_1(j))/(x1i(j)-x1i_1(j)+0.00001));
end
s1=sgn'*[r1(k),y1(k)];
wshi2_1=wshi(1:3,1:3);
alter=zeros(3,1);
dws1=zeros(3,2);
for j=1:1:3
    for p=1:1:3
        alter(j)=alter(j)+delta2(p,:)*wshi2_1(:,j);
    end
end
for p=1:1:3
    dws1(p,:)=alter(p)*s1(p,:);
end
w11=w11+rate1*dws1+rate3*(w11_1-w11_2);
```

```
%调整w12
for j=1:1:3
    sgn(j)=sign((h2i(j)-h2i_1(j))/(x2i(j)-x2i_1(j)+0.0000001));
end
s2=sgn'*[r2(k),y2(k)];
wshi2_2=wshi(:,4:6);
alter2=zeros(3,1);
dws2=zeros(3,2);
for j=1:1:3
    for p=1:1:3
        alter2(j)=alter2(j)+delta2(p,:)*wshi2_2(:,j);
    end
end
for p=1:1:3
    dws2(p,:)=alter2(p)*s2(p,:);
end
w12=w12+rate1*dws2+rate3*(w12_1-w12_2);
%调整w13
for j=1:1:3
    sgn(j)=sign((h3i(j)-h3i_1(j))/(x3i(j)-x3i_1(j)+0.0000001));
end
s3=sgn'*[r3(k),y3(k)];
wshi2_3=wshi(:,7:9);
alter3=zeros(3,1);
dws3=zeros(3,2);
for j=1:1:3
    for p=1:1:3
```

```
            alter3(j)=(alter3(j)+delta2(p,:)*wshi2_3(:,j));
        end
    end
    for p=1:1:3
        dws3(p,:)=alter2(p)*s3(p,:);
    end
    w13=w13+rate1*dws3+rate3*(w13_1-w13_2);
    %参数更新
    u_3=u_2;u_2=u_1;u_1=uh;
    y_2=y_1;y_1=yn;
    h1i_1=h1i;h2i_1=h2i;h3i_1=h3i;
    x1i_1=x1i;x2i_1=x2i;x3i_1=x3i;
    w11_1=w11;w11_2=w11_1;
    w12_1=w12;w12_2=w12_1;
    w13_1=w13;w13_2=w13_1;
    %第二层权值
    w21_1=w21;w21_2=w21_1;
    w22_1=w22;w22_2=w22_1;
    w23_1=w23;w23_2=w23_1;
end
% 结果分析
time=0.001*(1:k);
figure(1)
subplot(3,1,1)
plot(time,r1,'r-',time,y1,'b-');
title('PID神经元网络控制','fontsize',12);
ylabel('控制量1','fontsize',12);
```

```
legend('控制目标','实际输出','fontsize',12);
subplot(3,1,2)
plot(time,r2,'r-',time,y2,'b-');
ylabel('控制量2','fontsize',12);
legend('控制目标','实际输出','fontsize',12);
subplot(3,1,3)
plot(time,r3,'r-',time,y3,'b-');
xlabel('时间（秒）','fontsize',12);ylabel('控制量3','fontsize',12);
legend('控制目标','实际输出','fontsize',12);
figure(2)
plot(time,u1,'r-',time,u2,'g-',time,u3,'b');
title('PID神经网络提供给对象的控制输入');
xlabel('时间'),ylabel('被控量');
legend('u1','u2','u3');grid
figure(3)
plot(time,J,'r-');
axis([0,0.2,0,1]);grid
title('控制误差曲线','fontsize',12);
xlabel('时间','fontsize',12);ylabel('控制误差','fontsize',12);
```

注意：因为网络初始权值随机得到，所以每次运行的结果可能不一致。

运行以上代码，得到控制器控制效果如图16-22所示。

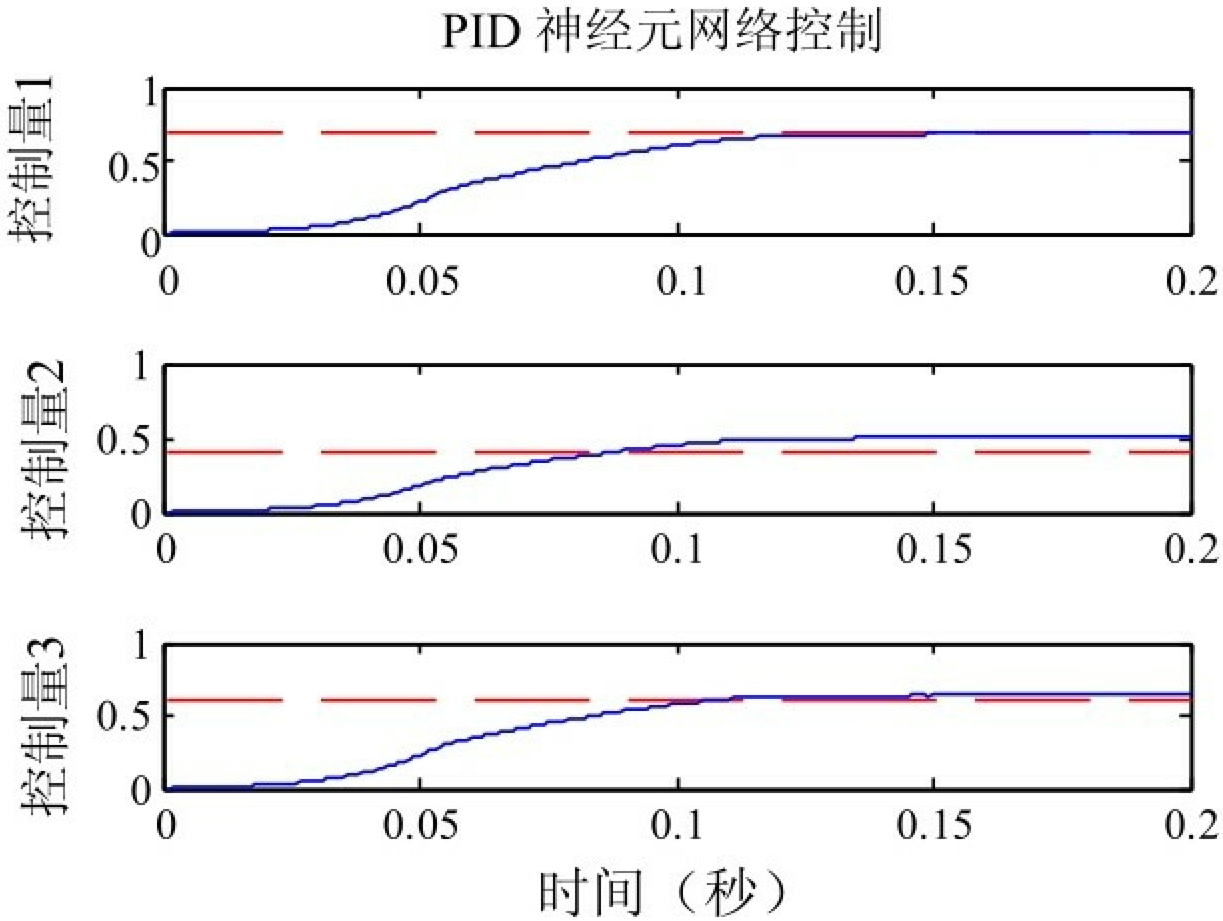


图16-22 控制器控制效果图

控制器的误差如图16-23所示。

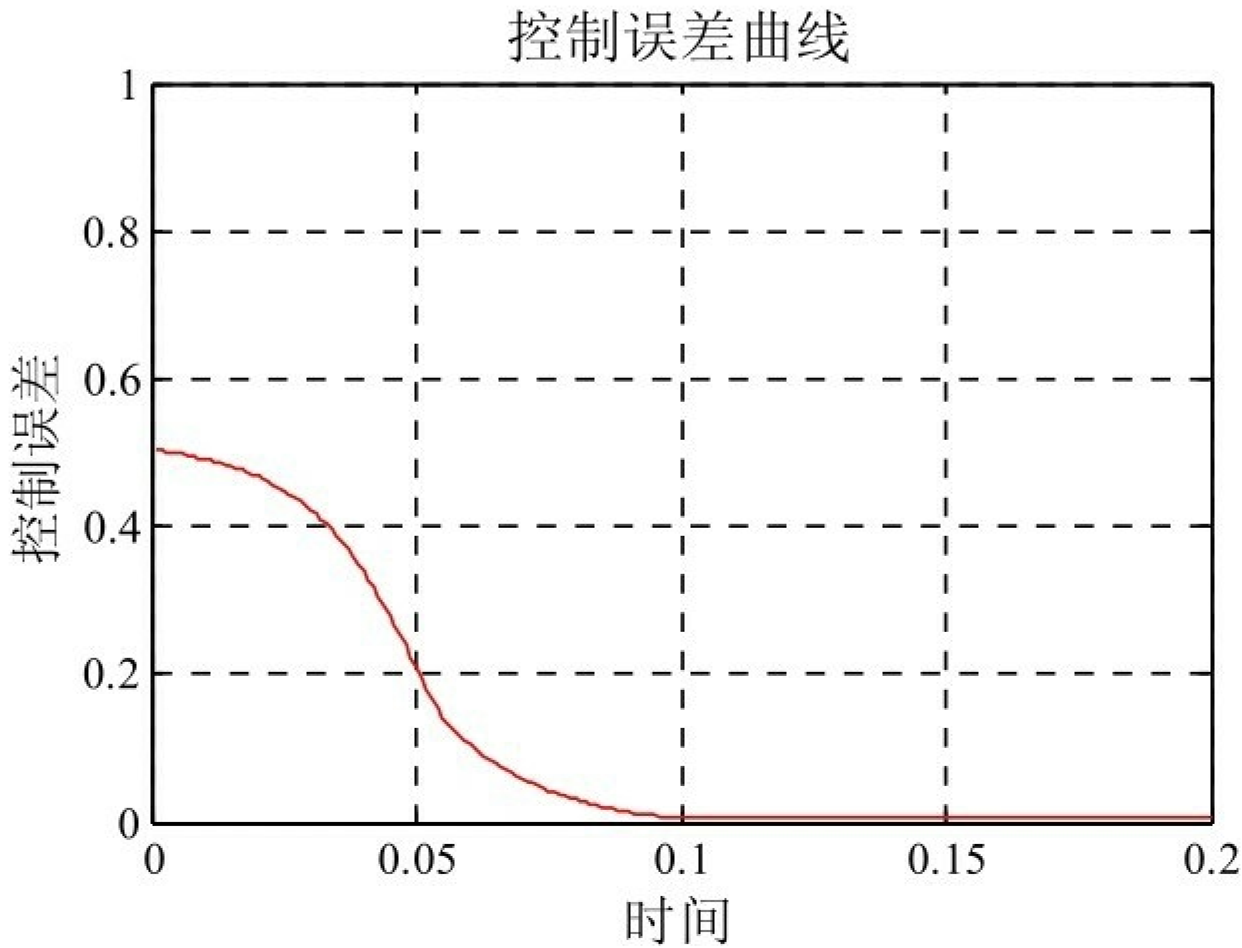


图16-23 控制器的误差

从图16-22和图16-23可以看出，PID神经元控制器能够很好地控制3输入3输出控制系统，控制量的最终值解决目标值。

在仿真过程中，连接权系数初值的选择对系统的调节过程有很大影响，应尽量取小些的值。另外，对于 BP 神经网络中隐含层节点的个数选择目前还没有理论指导依据，只能根据经验选取。

仿真结果表明，多输入多输出神经网络PID控制器可以在系统对象参数未知的情况下，通过自身的训练和学习，实现多变量系统的解耦控制，能够基本消除变量之间的耦合作用。

另外，控制器结构可以根据输入和输出变量的个数来确定，而不必预先知道控制对象的结构；网络连接权系数和PID参数的初值可以按照经典PID控制的经验来取值；通过调节网络阈值，可以调节系统的动态

和静态性能。

16.3 遗传算法优化神经网络

遗传算法（Genetic Algorithm, GA）最先是由美国Mic- hgan大学的John Holland于1975年提出的。遗传算法是模拟达尔文的遗传选择和自然淘汰的生物进化过程的计算模型。它的思想源于生物遗传学和适者生存的自然规律，是具有“生存+检测”的迭代过程的搜索算法。

遗传算法以一种群体中的所有个体为对象，并利用随机化技术指导对一个被编码的参数空间进行高效搜索。其中，选择、交叉和变异构成了遗传算法的遗传操作；参数编码、初始群体的设定、适应度函数的设计、遗传操作设计、控制参数设定这5个要素组成了遗传算法的核心内容。

遗传算法是一种基于生物自然选择与遗传机理的随机搜索算法，与传统搜索算法不同，遗传算法从一组随机产生的称为“种群（Population）”的初始解开始搜索过程。

遗传算法可用于分析神经网络。神经网络由于有分布存储等特点，一般难以从其拓扑结构直接理解其功能。遗传算法可对神经网络进行功能分析，性质分析，状态分析。

遗传算法虽然在多种领域都有实际应用，并且也展示了潜力和宽广前景；但是，遗传算法还有大量的问题需要研究，目前也还有各种不足。

首先，在变量多、取值范围大或无给定范围时，收敛速度下降；其次，可找到最优解附近，但无法精确确定最优解位置；最后，遗传算法的参数选择尚未有定量方法。对遗传算法，还需要进一步研究其数学基础理论；还需要在理论上证明它与其他优化技术的优劣及原因；还需研究硬件化的遗传算法；以及遗传算法的通用编程和形式等。

种群中的每个个体是问题的一个解，称为“染色体（chromosome）”。染色体是一串符号，比如一个二进制字符串。这些染色体在后续迭代中不断进化，称为遗传。在每一代中用“适值（fitness）”来测量染色体的好坏，生成的下一代染色体称为后代（offspring）。后代是由前一代染色体通过交叉（crossover）或者变异（mutation）运算形成的。

在新一代形成过程中，根据适度的大小选择部分后代，淘汰部分后代。从而保持种群大小是常数。适值高的染色体被选中的概率较高，这样经过若干代之后，算法收敛于最好的染色体，它很可能就是问题的最优解或次优解。主要步骤如下所示。

（1）编码。GA 在进行搜索之前先将解空间的解数据表示成遗传空间的基因型串结构数据，这些串结构数据的不同组合便构成了不同的点。

（2）初始群体的生成。随机产生N个初始串结构数据，每个串结构数据称为一个个体， N个个体构成了一个群体。GA以这N个串结构数据作为初始点开始迭代。

（3）适应性值评估检测。适应性函数表明个体或解的优劣性。对于不同的问题，适应性函数的定义方式也不同。

（4）选择。选择的目的是为了从当前群体中选出优良的个体 ，使它们有机会作为父代为下一代繁殖子孙。遗传算法通过选择过程体现这一思想，进行选择的原则是适应性强的个体为下一代贡献一个或多个后代的概率大。选择实现了达尔文的适者生存原则。

（5）交叉。交叉操作是遗传算法中最主要的遗传操作。通过交叉操作可以得到新一代个体，新个体组合了其父辈个体的特性。交叉体现了信息交换的思想。

（6）变异。变异首先在群体中随机选择一个个体，对于选中的个体以一定的概率随机地改变串结构数据中某个串的值。同生物界一样，

GA 中变异发生的概率很低，通常取值在 0.001～0.01 之间。变异为新个体的产生提供了机会。

实际上，遗传算法中有两类运算。

·遗传运算：交叉和变异。

·进化运算：选择。

遗传算法的特点分为以下几点。

·GA 是对问题参数的编码组进行计算，而不是针对参数本身。GA 的搜索是从问题解的编码组开始搜索、而不是从单个解开始。

·GA 使用目标函数值（适应度）这一信息进行搜索，而不需导数等其他信息。

·GA 算法使用的选择、交叉、变异这3 个算子都是随机操作，而不是确定规则。

下面举例说明遗传算法。

【例16-5】求函数f(x)=10·sin(5x)+7·cos(4x) x∈[0,10] 的最大值。

解：将x的值用一个10位的二值形式表示为二值问题，一个10位的二值数提供的分辨率是每位 (10-0)/(2^10-1)≈0.01 。

将变量域 [0,10] 离散化为二值域 [0,1023], x=0+10·b/1023, 其中b 是[0,1023]中的一个二值数。

（1）初始化（编码）。

initpop.m 函数的功能是实现群体的初始化，popsize 表示群体的大小，chromlength 表示染色体的长度（二值数的长度），长度大小取决于变量的二进制编码的长度（在本例中取10位）。

遗传算法子程序：初始化。

```
% rand随机产生每个单元为{0,1} 行数为popsize，列数为chromlength的矩阵，
% roud对矩阵的每个单元进行圆整。这样产生的初始种群。
function pop=initpop(popsize,chromlength)
```

% rand随机产生每个单元为{0,1} 行数为popsize，列数为chromlength的矩阵，

% roud对矩阵的每个单元进行圆整。这样产生的初始种群。

pop=round(rand(popsize,chromlength));

（2）计算目标函数值。

将二进制数转化为十进制数遗传算法子程序。

%产生 [2^n 2^(n-1) ...1] 的行向量，然后求和，将二进制转化为十进制

function pop2=decodebinary(pop)

%求pop行和列数

[px,py]=size(pop);

for i=1:py

pop1(:,i)=2.^(py-i).*pop(:,i);

end

%求pop1的每行之和

pop2=sum(pop1,2);

（3）将二进制编码转化为十进制数。

decodechrom.m 函数的功能是将染色体（或二进制编码）转换为十进制，参数 spoint表示待解码的二进制串的起始位置。

对于多个变量而言，如有两个变量，采用20位表示，每个变量10位，则第一个变量从1开始，另一个变量从11开始。

参数1ength表示所截取的长度（本例为10）。

遗传算法子程序如下。

%将二进制编码转换成十进制

function pop2=decodechrom(pop,spoint,length)

pop1=pop(:,spoint:spoint+length-1);

pop2=decodebinary(pop1);

（4）计算目标函数值。

calobjvalue 函数的功能是实现目标函数的计算，其公式采用本文示例仿真，可根据不同优化问题予以修改。

遗传算法子程序如下。

```
%实现目标函数的计算
function [objvalue]=calobjvalue(pop)
temp1=decodechrom(pop,1,10);%将pop每行转化成十进制数
x=temp1*10/1023;%将二值域中的数转化为变量域的数
objvalue=10*sin(5*x)+7*cos(4*x);%计算目标函数值
```

（5）计算个体的适应值。

遗传算法子程序如下。

```
%计算个体的适应值
function fitvalue=calfitvalue(objvalue)
global Cmin;
Cmin=0;
[px,py]=size(objvalue);
for i=1:px
if objvalue(i)+Cmin>0
temp=Cmin+objvalue(i);
else
temp=0.0;
end
fitvalue(i)=temp;
end
fitvalue=fitvalue';
```

（6）选择复制。

选择或复制操作是决定哪些个体可以进入下一代。程序中采用赌轮

盘选择法选择，这种方法较易实现。

遗传算法子程序如下所示。

```
%选择复制
function [newpop]=selection(pop,fitvalue)
totalfit=sum(fitvalue);%求适应值之和
fitvalue=fitvalue/totalfit;%单个个体被选择的概率
%如 fitvalue=[1 2 3 4]，则 cumsum(fitvalue)=[1 3 6 10]
%[px,py]=size(pop);
fitvalue=cumsum(fitvalue);
ms=sort(rand(px,1));%从小到大排列
fitin=1;
newin=1;
while newin<=px
if(ms(newin))<fitvalue(fitin)
newpop(newin)=pop(fitin);
newin=newin+1;
else
fitin=fitin+1;
end
end
```

（7）交叉。

交叉（crossover），群体中的每个个体之间都以一定的概率 pc 交叉，即两个个体从各自字符串的某一位置（一般是随机确定）开始互相交换，这类似生物进化过程中的基因分裂与重组。例如，假设2个父代个体x1，x2为

x1=0100110

x2=1010001

从每个个体的第3位开始交叉，交叉后得到2个新的子代个体y1，y2分别为

y1＝0100001

y2＝1010110

这样2个子代个体就分别具有了2个父代个体的某些特征。利用交叉我们有可能由父代个体在子代组合成具有更高适合度的个体。

事实上交叉是遗传算法区别于其他传统优化方法的主要特点之一。

交叉遗传算法子程序如下。

```
function [newpop]=crossover(pop,pc)
[px,py]=size(pop);
newpop=ones(size(pop));
for i=1:2:px-1
if(rand<pc)
cpoint=round(rand*py);
newpop(i,:)=[pop(i,1:cpoint),pop(i+1,cpoint+1:py)];
newpop(i+1,:)=[pop(i+1,1:cpoint),pop(i,cpoint+1:py)];
else
newpop(i,:)=pop(i);
newpop(i+1,:)=pop(i+1);
end
end
```

（8）变异。

变异（mutation），基因的突变普遍存在于生物的进化过程中。变异是指父代中的每个个体的每一位都以概率 pm 翻转，即由“1”变为“0”，或由“0”变为“1”。遗传算法的变异特性可以使求解过程随机地搜索到解可能存在的整个空间，因此可以在一定程度上求得全局最优解。

遗传算法子程序如下。

```
%变异
function [newpop]=mutation(pop,pm)
[px,py]=size(pop);
newpop=ones(size(pop));
for i=1:px
if(rand<pm)
mpoint=round(rand*py);
if mpoint<=0
mpoint=1;
end
newpop(i)=pop(i);
if any(newpop(i,mpoint))==0
newpop(i,mpoint)=1;
else
newpop(i,mpoint)=0;
end
else
newpop(i)=pop(i);
end
end
```

（9）求出群体中最大的适应值及其个体。

遗传算法子程序如下所示。

```
%求出群体中适应值最大的值
function [bestindividual,bestfit]=best(pop,fitvalue)
[px,py]=size(pop);
bestindividual=pop(1,:);
bestfit=fitvalue(1);
```

```
for i=2:px
if fitvalue(i)>bestfit
bestindividual=pop(i,:);
bestfit=fitvalue(i);
end
end
```

（10）主程序。

遗传算法主程序如下所示。

```
clear all
clf
popsize=20;%群体大小
chromlength=10;%字符串长度（个体长度）
pc=0.6;%交叉概率
pm=0.001;%变异概率
pop=initpop(popsize,chromlength);   %随机产生初始群体
for i=1:20 %20为迭代次数
[objvalue]=calobjvalue(pop);       %计算目标函数
fitvalue=calfitvalue(objvalue);       %计算群体中每个个体的适应度
[newpop]=selection(pop,fitvalue);       %复制
[newpop]=crossover(pop,pc);       %交叉
[newpop]=mutation(pop,pc);       %变异
[bestindividual,bestfit]=best(pop,fitvalue);   %求出群体中适应值最大的个体及其适应值
y(i)=max(bestfit);
n(i)=i;
pop5=bestindividual;
x(i)=decodechrom(pop5,1,chromlength)*10/1023;
```

pop=newpop;

end

fplot('10*sin(5*x)+7*cos(4*x)',[0 10])

hold on

plot(x,y,'r*')

hold off

%计算最大值及其位置

[z index]=max(y);

%计算最大值对应的x值

x5=x(index)

y=z

运行上述程序得到以下结果。

>> x

x =

Columns 1 through 9

1.5640 9.3744 9.3744 9.3744 4.3695 9.3744 4.3695
4.3695 4.9951

Columns 10 through 18

9.3744 4.9951 9.3744 9.3744 9.3744 9.3744 4.3695
9.3744 4.3695

Columns 19 through 20

9.3744 4.3695

>> index

index =

1

>>x5

x5 =

```
    1.5640
>> y
y =
    16.9917
```

由index=1可知，在x第一个点（即1.5640）位置上，y值最大。

用遗传算法得到的仿真点和目标曲线如图 16-24 所示。从图中可以看出，在 x=1.564时，函数f(x)有最大值为16.9917。

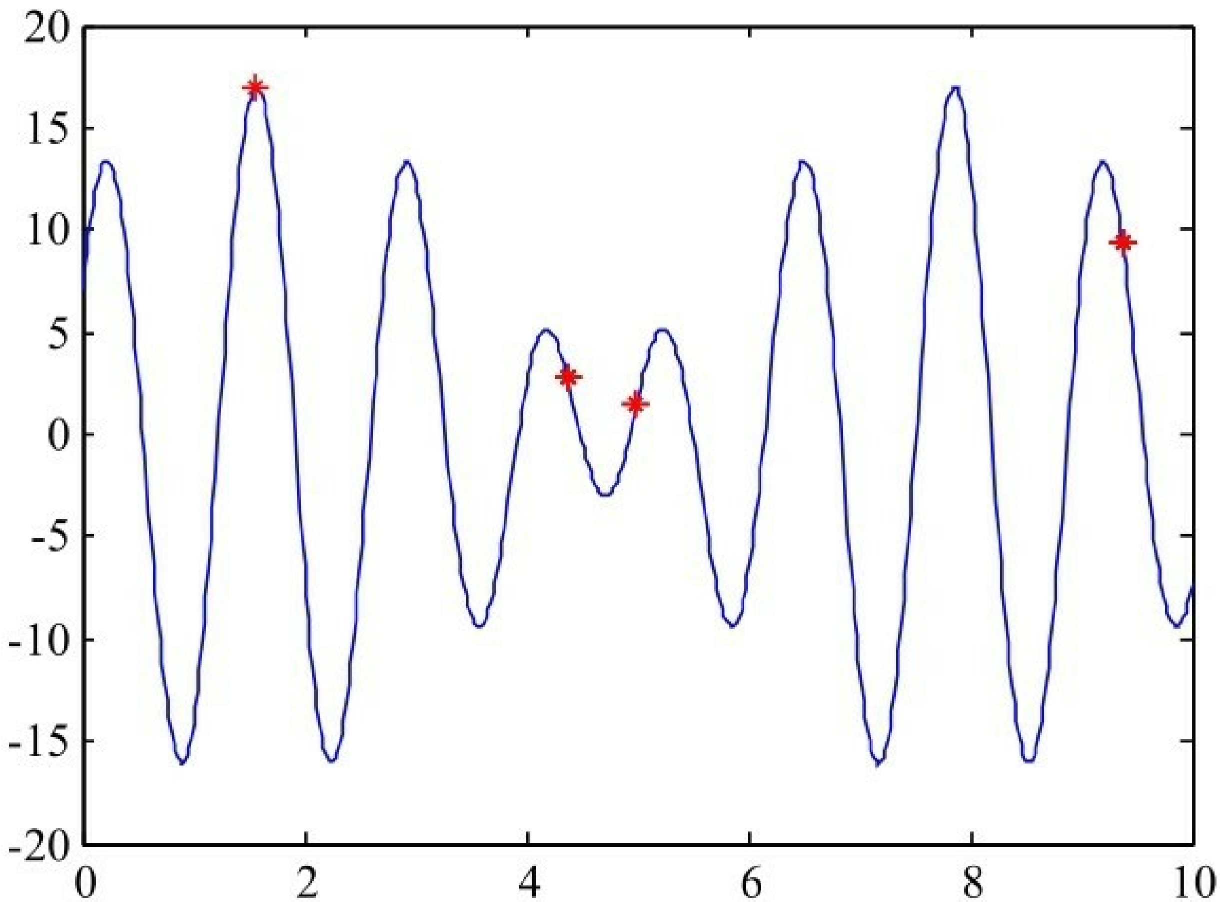


图16-24 目标曲线和用遗传算法得到的仿真点图形

16.4 模糊神经网络控制

美国加州大学的L.A.Zadeh教授在1965年发表了著名的论文，文中首次提出表达事物模糊性的重要概念——隶属函数，从而突破了19世纪末笛卡尔的经典集合理论，奠定模糊理论的基础。

模糊理论和神经网络技术是近年来人工智能研究较为活跃的两个领域。人工神经网络是模拟人脑结构的思维功能，具有较强的自学习和联想功能，人工干预少，精度较高，对专家知识的利用也较少。

模糊神经网络有如下3种形式。

·逻辑模糊神经网络。

·算术模糊神经网络。

·混合模糊神经网络。

模糊神经网络就是具有模糊权系数或者输入信号是模糊量的神经网络。上面3种形式的模糊神经网络中所执行的运算方法不同。

模糊神经网络无论作为逼近器，还是模式存储器，都是需要学习和优化权系数的。学习算法是模糊神经网络优化权系数的关键。

对于逻辑模糊神经网络，可采用基于误差的学习算法，也即是监视学习算法。对于算术模糊神经网络，则有模糊BP算法和遗传算法等。

对于混合模糊神经网络，目前尚未有合理的算法；不过，混合模糊神经网络一般是用于计算而不是用于学习的，它不必一定学习。

一种基于T—S 模型的模糊神经网络由前件网络和后件网络两部分组成。前件网络用来匹配模糊规则的前件，它相当于每条规则的适用度。后件网络用来实现模糊规则的后件。总的输出为各模糊规则后件的加权和，加权系数为各条规则的适用度。

模糊神经网络具有局部逼近功能，且具有神经网络和模糊逻辑两者

的优点。它既可以容易地表示模糊和定性的知识，又具有较好的学习能力。

水质评价指按照评价目标，选择相应的水质参数、水质标准和评价方法，对水体的质量、利用价值及水的处理要求作出评定。水质评价是合理开发利用和保护水资源的一项基本工作。根据不同评价类型，采用相应的水质标准。

评价水环境质量，采用地面水环境质量标准；评价养殖水体的质量，采用渔业用水水质标准；评价集中式生活饮用水取水点的水源水质，用地面水卫生标准；评价农田灌溉用水，采用农田灌溉水质标准。

一般都以国家或地方政府颁布的各类水质标准作为评价标准。在无规定水质标准情况下，可采用水质基准或本水系的水质背景值作为评价标准。

现采取江水样本对江水水质进行评价，采取的取水口分别记为A，B和C厂。三个水厂水中的氨氧含量变化趋势如图16-25～图16-27所示。

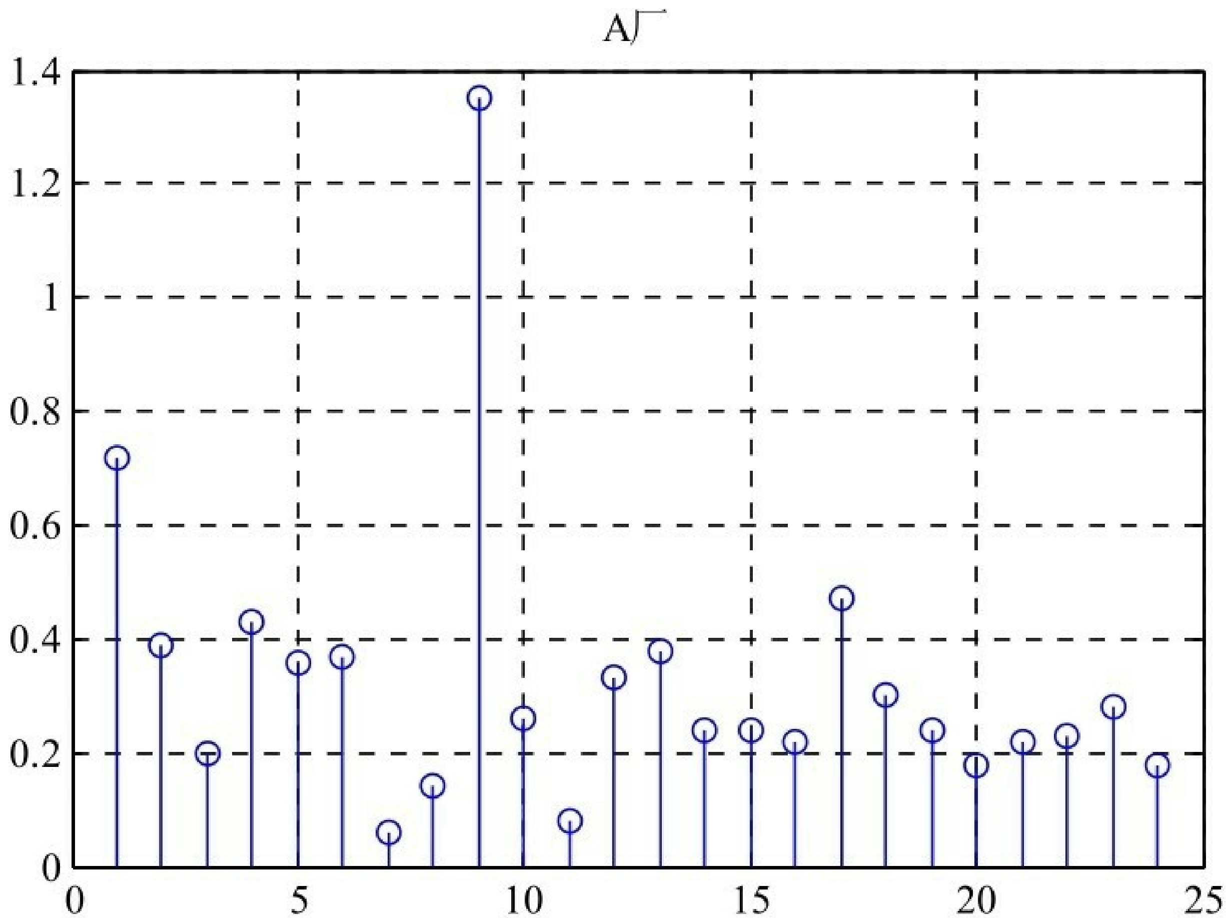


图16-25 A厂水中的氨氧含量变化趋势

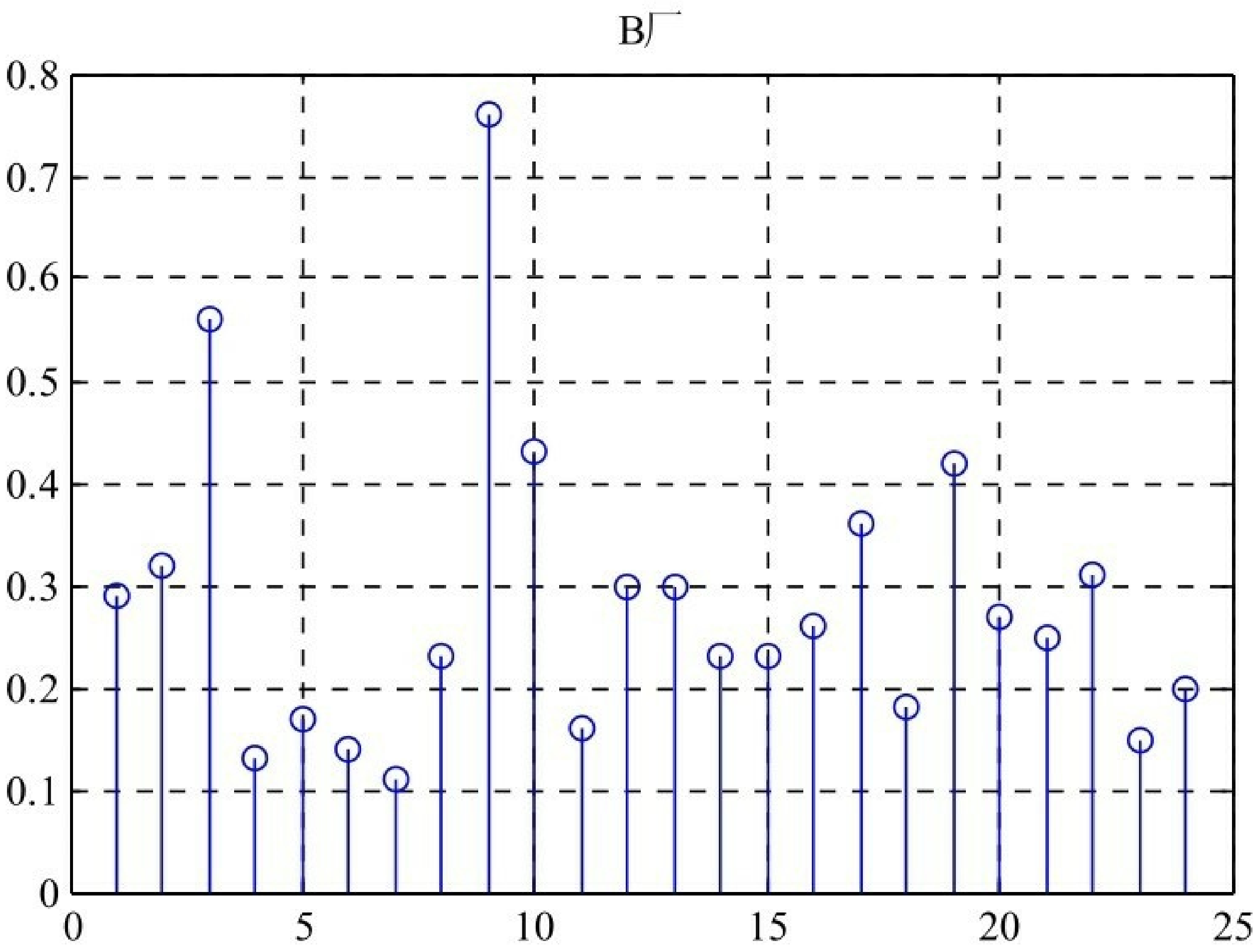


图16-26 B厂水中的氨氧含量变化趋势

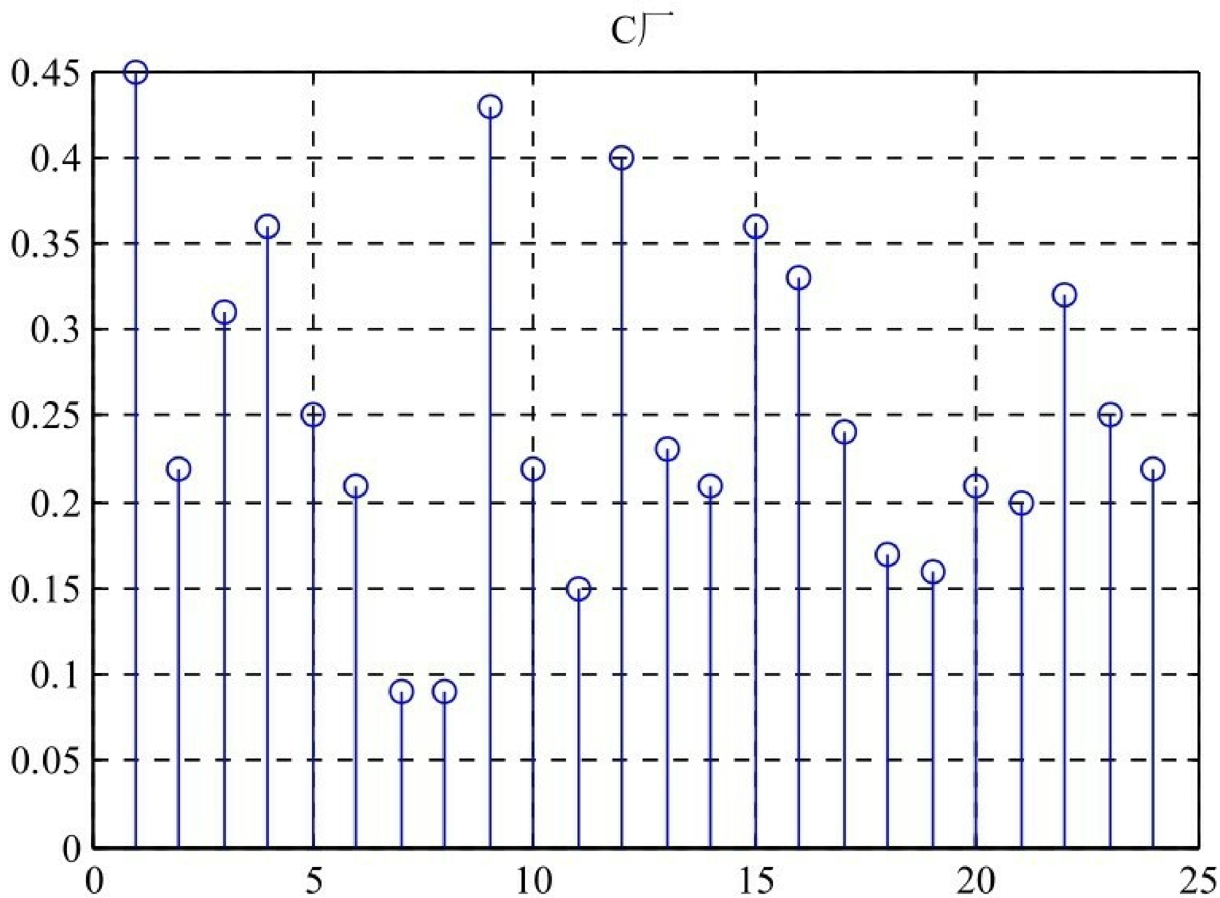


图16-27 C厂水中的氨氧含量变化趋势

从图16-25～图16-27可以看出，C厂水中的氨氧含量低于A和B厂的。

【例16-6】应用模糊神经网络算法，实现江水水质的评价。

解：根据训练输入/输出数据维数确定网络结构，初始化模糊神经网络隶属于函数参数和系数，归一化训练数据。

从数据库文件 data1 中导出数据。其中，因为江水水质评价的真实数据比较难找，文中随机给出5类数据作为江水水质评价的5种因素。

MATLAB程序如下所示。

```
clc
clear
% 参数初始化
```

```
xite=0.002;
alfa=0.04;
%网络节点
I=6; %输入节点数
M=10; %隐含节点数
O=1; %输出节点数
%系数初始化
p0=0.3*ones(M,1);p0_1=p0;p0_2=p0_1;
p1=0.3*ones(M,1);p1_1=p1;p1_2=p1_1;
p2=0.3*ones(M,1);p2_1=p2;p2_2=p2_1;
p3=0.3*ones(M,1);p3_1=p3;p3_2=p3_1;
p4=0.3*ones(M,1);p4_1=p4;p4_2=p4_1;
p5=0.3*ones(M,1);p5_1=p5;p5_2=p5_1;
p6=0.3*ones(M,1);p6_1=p6;p6_2=p6_1;
%参数初始化
c=1+rands(M,I);c_1=c;c_2=c_1;
b=1+rands(M,I);b_1=b;b_2=b_1;
maxgen=120;%进化次数
%网络测试数据，并对数据归一化
load data1 input_train output_train input_test output_test
%选连样本输入输出数据归一化
[inputn,inputps]=mapminmax(input_train);
[outputn,outputps]=mapminmax(output_train);
[n,m]=size(input_train);
%%%%%%% 网络训练%%%%%%%%%%%
%循环开始，进化网络
for iii=1:maxgen
```

```
iii
for k=1:m
    x=inputn(:,k);
    %输出层结算
    for i=1:I
        for j=1:M
            u(i,j)=exp(-(x(i)-c(j,i))^2/b(j,i));
        end
    end
    %模糊规则计算
    for i=1:M
        w(i)=u(1,i)*u(2,i)*u(3,i)*u(4,i)*u(5,i)*u(6,i);
    end
    addw=sum(w);
    for i=1:M
        yi(i)=p0_1(i)+p1_1(i)*x(1)+p2_1(i)*x(2)+p3_1(i)*x(3)+p4_1(i)*
    end
    addyw=yi*w';
    %网络预测计算
    yn(k)=addyw/addw;
    e(k)=outputn(k)-yn(k);
    %计算p的变化值
    d_p=zeros(M,1);
    d_p=xite*e(k)*w./addw;
    d_p=d_p';
    %计算b变化值
    d_b=0*b_1;
```

```
for i=1:M
    for j=1:I
        d_b(i,j)=xite*e(k)*(yi(i)*addw-addyw)*(x(j)-
c(i,j))^2*w(i)/(b(i,j)^2*addw^2);
    end
end
%更新c变化值
for i=1:M
    for j=1:I
        d_c(i,j)=xite*e(k)*(yi(i)*addw-addyw)*2*(x(j)-
c(i,j))*w(i)/(b(i,j)*addw^2);
    end
end
p0=p0_1+ d_p+alfa*(p0_1-p0_2);
p1=p1_1+ d_p*x(1)+alfa*(p1_1-p1_2);
p2=p2_1+ d_p*x(2)+alfa*(p2_1-p2_2);
p3=p3_1+ d_p*x(3)+alfa*(p3_1-p3_2);
p4=p4_1+ d_p*x(4)+alfa*(p4_1-p4_2);
p5=p5_1+ d_p*x(5)+alfa*(p5_1-p5_2);
p6=p6_1+ d_p*x(6)+alfa*(p6_1-p6_2);
b=b_1+d_b+alfa*(b_1-b_2);
c=c_1+d_c+alfa*(c_1-c_2);
p0_2=p0_1;p0_1=p0;
p1_2=p1_1;p1_1=p1;
p2_2=p2_1;p2_1=p2;
p3_2=p3_1;p3_1=p3;
p4_2=p4_1;p4_1=p4;
```

```
        p5_2=p5_1;p5_1=p5;
        p6_2=p6_1;p6_1=p6;
        c_2=c_1;c_1=c;
        b_2=b_1;b_1=b;
    end
    E(iii)=sum(abs(e));
end
figure(1);
plot(outputn,'r')
hold on
plot(yn,'b')
hold on
plot(outputn-yn,'g');
legend('实际输出','预测输出','误差','fontsize',12)
title('训练数据预测','fontsize',12)
xlabel('样本序号','fontsize',12)
ylabel('水质等级','fontsize',12)
%%%%%%%%%%% 网络预测%%%%%%%%%%%%%%%%%%
%数据归一化处理
inputn_test=mapminmax('apply',input_test,inputps);
[n,m]=size(inputn_test)
for k=1:m
    x=inputn_test(:,k);
    %计算输出中间层
    for i=1:I
        for j=1:M
            u(i,j)=exp(-(x(i)-c(j,i))^2/b(j,i));
```

```
        end
    end
    for i=1:M
        w(i)=u(1,i)*u(2,i)*u(3,i)*u(4,i)*u(5,i)*u(6,i);
    end
    addw=0;
    for i=1:M
        addw=addw+w(i);
    end
    for i=1:M
        yi(i)=p0_1(i)+p1_1(i)*x(1)+p2_1(i)*x(2)+p3_1(i)*x(3)+p4_1(i)*x(
    end
    addyw=0;
    for i=1:M
        addyw=addyw+yi(i)*w(i);
    end
    %计算输出
    yc(k)=addyw/addw;
end
%%%%%%%%%%预测结果反归一
化%%%%%%%%%%%%%%%%%%%%%%%
test_simu=mapminmax('reverse',yc,outputps);
%作图
figure(2)
plot(output_test,'r')
hold on
plot(test_simu,'b')
```

```
hold on
plot(test_simu-output_test,'g')
legend('实际输出','预测输出','误差','fontsize',12)
title('测试数据预测','fontsize',12)
xlabel('样本序号','fontsize',12)
ylabel('水质等级','fontsize',12)
%%%%%%%%%%%%%%%江水实际水质预测%%%%%%%%%%%%%%%%%%%%%%
load data2 C B A
%%%%%%%%%%%%%%%%%%%%%%%%%C厂%%%%%%%%%%%%%%%%%%%%%%%%%%
zssz=C;
%数据归一化
inputn_test =mapminmax('apply',zssz,inputps);
[n,m]=size(zssz);
for k=1:1:m
  x=inputn_test(:,k);
  %计算输出中间层
  for i=1:I
    for j=1:M
      u(i,j)=exp(-(x(i)-c(j,i))^2/b(j,i));
    end
  end
  for i=1:M
    w(i)=u(1,i)*u(2,i)*u(3,i)*u(4,i)*u(5,i)*u(6,i);
  end
  addw=0;
```

```
    for i=1:M
        addw=addw+w(i);
    end
    for i=1:M
        yi(i)=p0_1(i)+p1_1(i)*x(1)+p2_1(i)*x(2)+p3_1(i)*x(3)+p4_1(i)*x(
    end
    addyw=0;
    for i=1:M
        addyw=addyw+yi(i)*w(i);
    end
    %计算输出
    szzb(k)=addyw/addw;
end
szzbz1=mapminmax('reverse',szzb,outputps);
for i=1:m
    if szzbz1(i)<=1.5
        szpj1(i)=1;
    elseif szzbz1(i)>1.5&&szzbz1(i)<=2.5
        szpj1(i)=2;
    elseif szzbz1(i)>2.5&&szzbz1(i)<=3.5
        szpj1(i)=3;
    elseif szzbz1(i)>3.5&&szzbz1(i)<=4.5
        szpj1(i)=4;
    else
        szpj1(i)=5;
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%B
厂%%%%%%%%%%%%%%%%%%%%%%%%
zssz=B;
inputn_test =mapminmax('apply',zssz,inputps);
[n,m]=size(zssz);
for k=1:1:m
    x=inputn_test(:,k);
    %计算输出中间层
    for i=1:I
        for j=1:M
            u(i,j)=exp(-(x(i)-c(j,i))^2/b(j,i));
        end
    end
    for i=1:M
        w(i)=u(1,i)*u(2,i)*u(3,i)*u(4,i)*u(5,i)*u(6,i);
    end
    addw=0;
    for i=1:M
        addw=addw+w(i);
    end
    for i=1:M
        yi(i)=p0_1(i)+p1_1(i)*x(1)+p2_1(i)*x(2)+p3_1(i)*x(3)+p4_1(i)*x(
    end
    addyw=0;
    for i=1:M
        addyw=addyw+yi(i)*w(i);
    end
```

```
    %计算输出
    szzb(k)=addyw/addw;
end
szzbz2=mapminmax('reverse',szzb,outputps);
for i=1:m
    if szzbz2(i)<=1.5
        szpj2(i)=1;
    elseif szzbz2(i)>1.5&&szzbz2(i)<=2.5
        szpj2(i)=2;
    elseif szzbz2(i)>2.5&&szzbz2(i)<=3.5
        szpj2(i)=3;
    elseif szzbz2(i)>3.5&&szzbz2(i)<=4.5
        szpj2(i)=4;
    else
        szpj2(i)=5;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%大溪沟水厂%%%%%%%%%%%%%%%%%%%%%%%%%
zssz=A;
inputn_test =mapminmax('apply',zssz,inputps);
[n,m]=size(zssz);
for k=1:1:m
    x=inputn_test(:,k);
    %计算输出中间层
    for i=1:I
        for j=1:M
```

```
            u(i,j)=exp(-(x(i)-c(j,i))^2/b(j,i));
        end
    end
    for i=1:M
        w(i)=u(1,i)*u(2,i)*u(3,i)*u(4,i)*u(5,i)*u(6,i);
    end
    addw=0;
    for i=1:M
        addw=addw+w(i);
    end
    for i=1:M
        yi(i)=p0_1(i)+p1_1(i)*x(1)+p2_1(i)*x(2)+p3_1(i)*x(3)+p4_1(i)*x(
    end
    addyw=0;
    for i=1:M
        addyw=addyw+yi(i)*w(i);
    end
    %计算输出
    szzb(k)=addyw/addw;
end
szzbz3=mapminmax('reverse',szzb,outputps);
for i=1:m
    if szzbz3(i)<=1.5
        szpj3(i)=1;
    elseif szzbz3(i)>1.5&&szzbz3(i)<=2.5
        szpj3(i)=2;
    elseif szzbz3(i)>2.5&&szzbz3(i)<=3.5
```

```
        szpj3(i)=3;
    elseif szzbz3(i)>3.5&&szzbz3(i)<=4.5
        szpj3(i)=4;
    else
        szpj3(i)=5;
    end
end
figure(3)
plot(szzbz1,'o-r')
hold on
plot(szzbz2,'*-g')
hold on
plot(szzbz3,'*:b')
xlabel('时间','fontsize',12)
ylabel('预测水质','fontsize',12)
legend('C','B','A','fontsize',12)
```

运行上述程序，得到训练数据预测结果如图16-28所示，测试数据预测如图16-29所示。

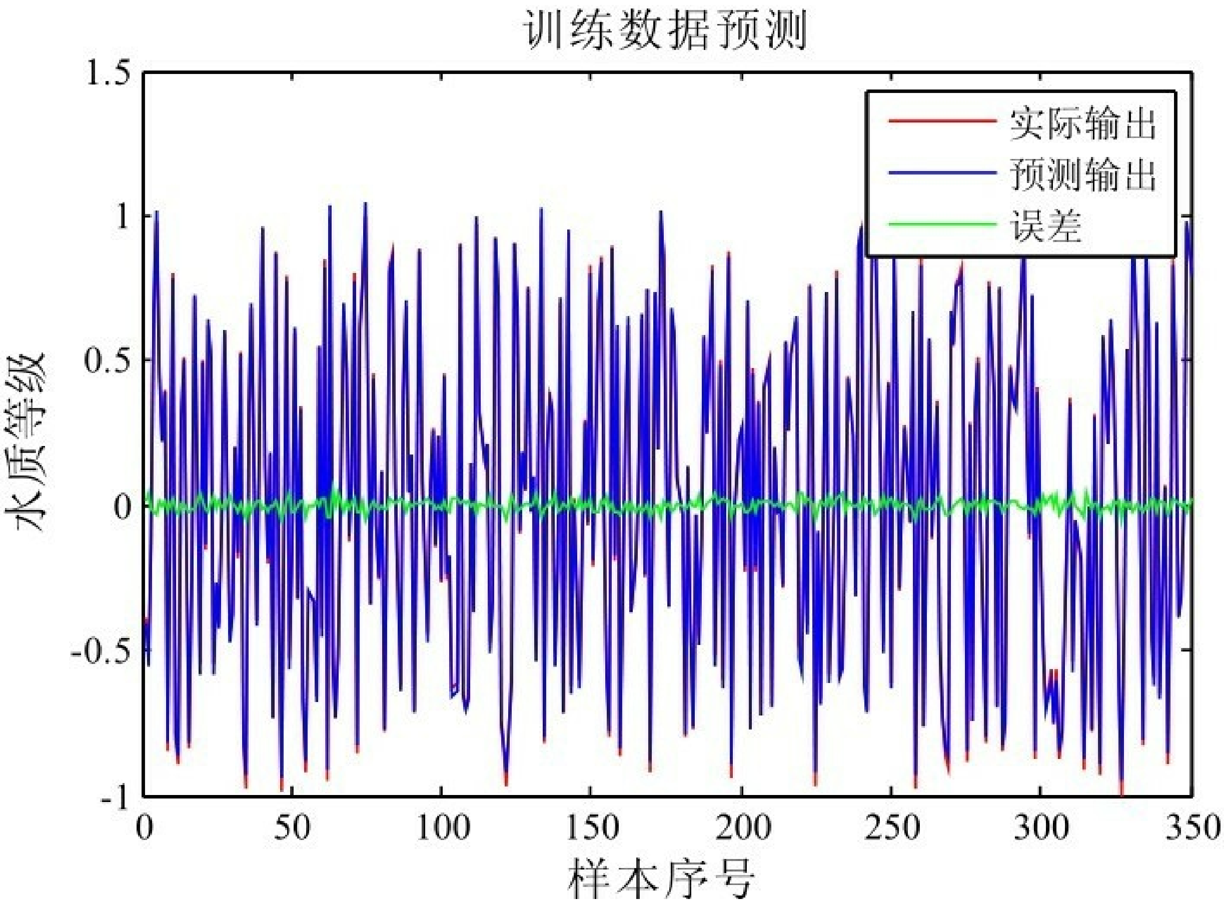


图16-28 训练数据预测结果

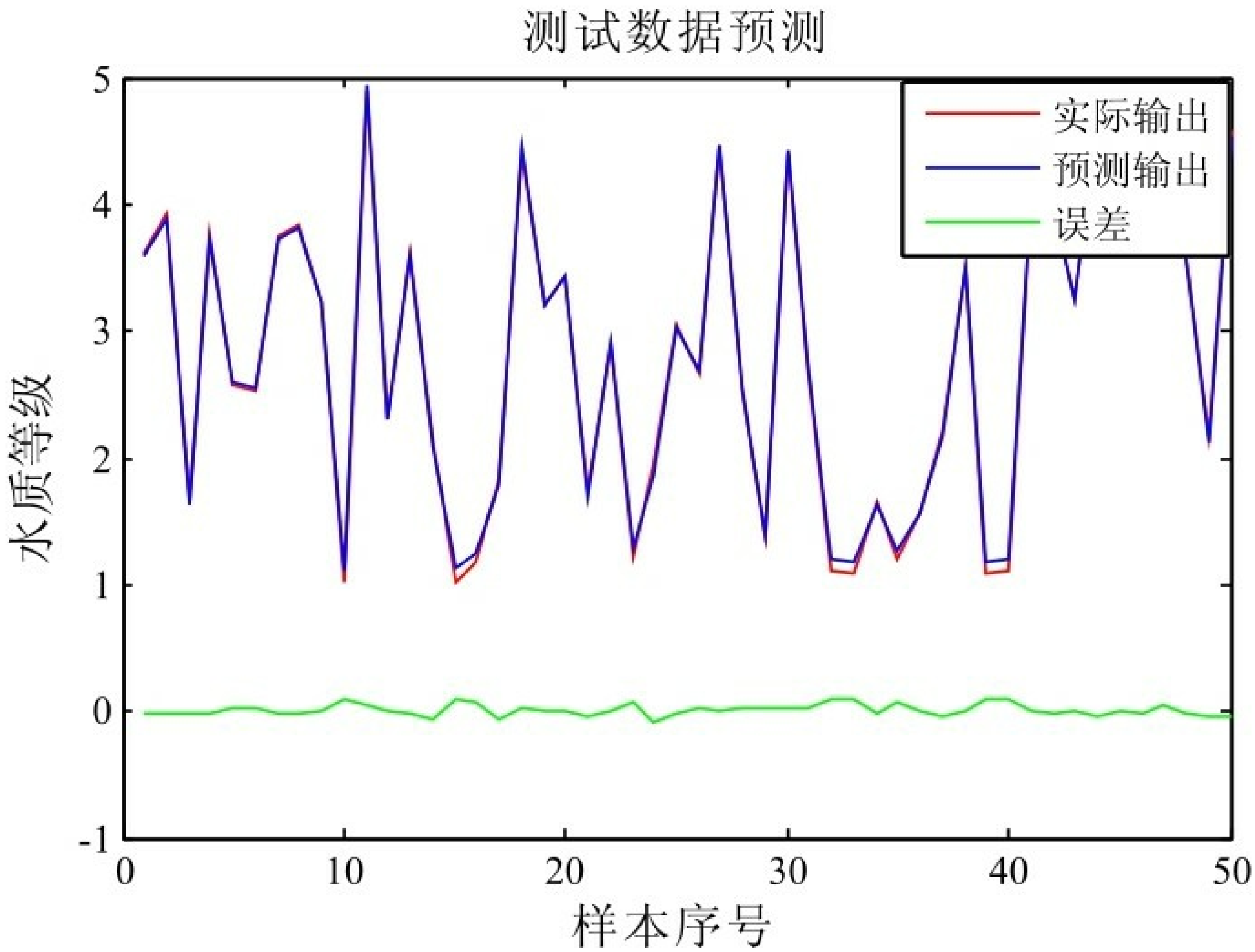


图16-29 测试数据预测

得到模糊神经网络对A、B和C厂的水质评价如图16-30所示，图中横坐标0～25表示从开始计算水质的月份到之后的第25个计算的水质月份。

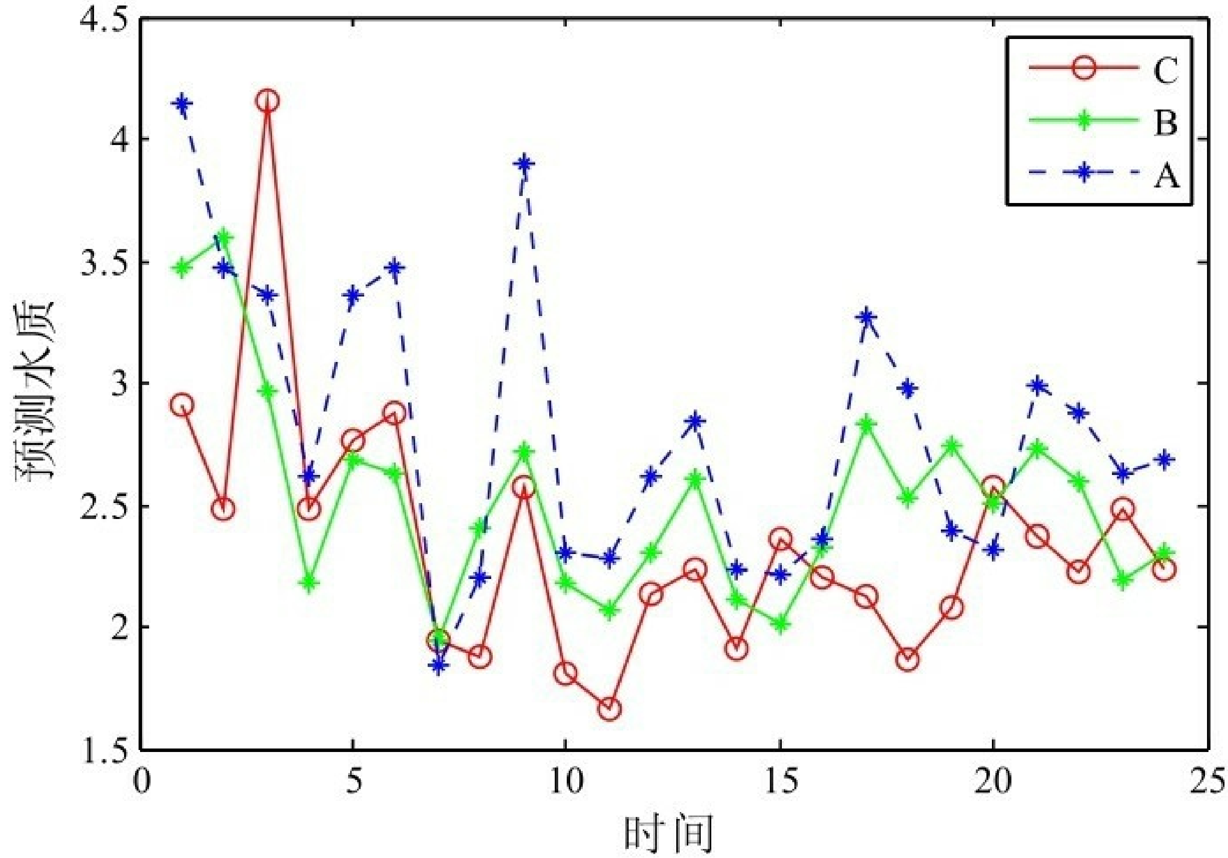


图16-30 模糊神经网络对A、B和C厂的水质评价

从图中可以看出A厂水质要好于B和C厂水质，这与前面有关氨氧含量的比较结果相符，这说明了模糊网络预测结果的有效性。

16.5 概率神经网络分类预测

概率神经网络是由 Specht 博士在1989 年提出的，它与统计信号处理的许多概念有着紧密的联系。当这种网络用于检测和模式分类时，可以得到贝叶斯最优结果。

概率神经网络通常由4层组成，其结构图如图16-31所示。

第一层为输入层，每个神经元均为单输入单输出，其传递函数也为线性的，这一层的作用只是将输入信号用分布的方式来表示。第二层称之为模式层，它与输入层之间通过连接权值W_{ij}相连接，模式层神经元的传递函数不再是通常的Sigmoid函数，而是如下所示。

g(Zi)=exp[(Zi-1)/(s*s)]

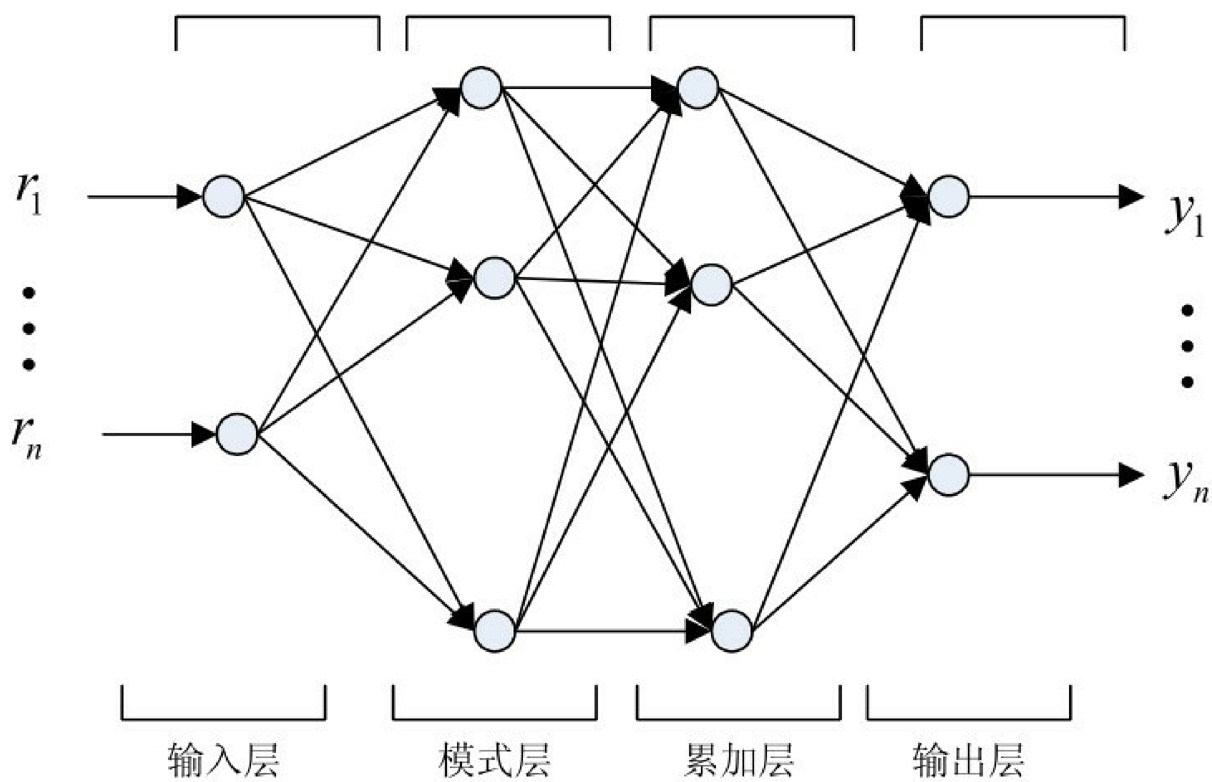


图16-31 概率神经网络

其中，Zi为该层第i个神经元的输入，s为均方差。第三层称之为累加层，它具有线性求和的功能。这一层的神经元数目与欲分的模式数目相同。第四层即输出层具有判决功能，它的神经元输出为离散值1和−1（或0），分别代表着输入模式的类别。

许多研究已表明概率神经网络具有如下特性。

·训练容易，收敛速度快，从而非常适用于实时处理。

·可以完成任意的非线性变换，所形成的判决曲面与贝叶斯最优准则下的曲面相接近。

·具有很强的容错性。

·模式层的传递函数可以选用各种用来估计概率密度的核函数，并且，分类结果对核函数的形式不敏感。

·各层神经元的数目比较固定，因而易于硬件实现。

故障诊断是利用各种检查和测试方法，发现系统和设备是否存在故障的过程是故障检测；而进一步确定故障所在大致部位的过程是故障定位。故障检测和故障定位同属网络生存性范畴。要求把故障定位到实施修理时可更换的产品层次（可更换单位）的过程为故障隔离。故障诊断就是指故障检测和故障隔离的过程。

概率说明网络结构简单，训练迅速，利用其强大的非线性分类能力，可以非常准确地完成故障诊断。

创建概率神经网络的函数是newpnn。概率神经网络是一种适用于分类问题的径向基网络，其调用格式如下。

net = newpnn(P,T,spread)

其中，P为输入向量，T为目标向量，spread为径向基函数的扩展速度参数。

例如有输入和目标向量如下。

P = [1 2 3 4 5 6 7];

Tc = [1 2 3 2 2 3 1];

如果要对上述向量进行故障诊断，则可以通过以下代码实现。

```
T = ind2vec(Tc)
net = newpnn(P,T);
Y = sim(net,P)
Yc = vec2ind(Y)
```

运行代码得到结果如下。

```
>> Y =
   1    0    0    0    0    0    1
   0    1    0    1    1    0    0
   0    0    1    0    0    1    0
>> Yc =
   1    2    3    2    2    3    1
```

由此可知，概率神经网络非常准确地判断出了故障的位置。

下面介绍使用概率神经网络建立故障诊断模型的案例。

【例16-7】根据给定的数据，建立基于概率神经网络的故障诊断模型。

解：应用newpnn函数创建概率神经网络，具体MATLAB代码如下所示。

```
clc;
clear all
nntwarn off;
warning off;
%数据载入
load data1
%选取训练数据和测试数据
Train=data(1:40,:);
Test=data(41:end,:);
```

```
p_train=Train(:,1:3)';
t_train=Train(:,4)';
p_test=Test(:,1:3)';
t_test=Test(:,4)';
% 将期望类别转换为向量
t_train=ind2vec(t_train);
t_train_temp=Train(:,4)';
% 使用newpnn函数建立PNN SPREAD选取为1.5
Spread=1.5;
net=newpnn(p_train,t_train,Spread)
% 训练数据回代查看网络的分类效果
% Sim函数进行网络预测
Y=sim(net,p_train);
% 将网络输出向量转换为指针
Yc=vec2ind(Y);
% 通过作图观察网络对训练数据分类效果
figure(1)
subplot(1,2,1)
stem(1:length(Yc),Yc,'bo')
hold on
stem(1:length(Yc),t_train_temp,'r*')
title('训练后的效果')
xlabel('样本编号')
ylabel('分类结果')
set(gca,'Ytick',[1:5])
subplot(1,2,2)
H=Yc-t_train_temp;
```

```
stem(H)
title('训练后的误差图')
xlabel('样本编号')
% 网络预测未知数据效果
Y2=sim(net,p_test);
Y2c=vec2ind(Y2)
figure(2)
stem(1:length(Y2c),Y2c,'b^')
hold on
stem(1:length(Y2c),t_test,'r*')
title('PNN 网络的预测效果')
xlabel('预测样本编号')
ylabel('分类结果')
set(gca,'Ytick',[1:5])
```

运行以上代码得到训练后训练数据网络的分类效果如图16-32所示。

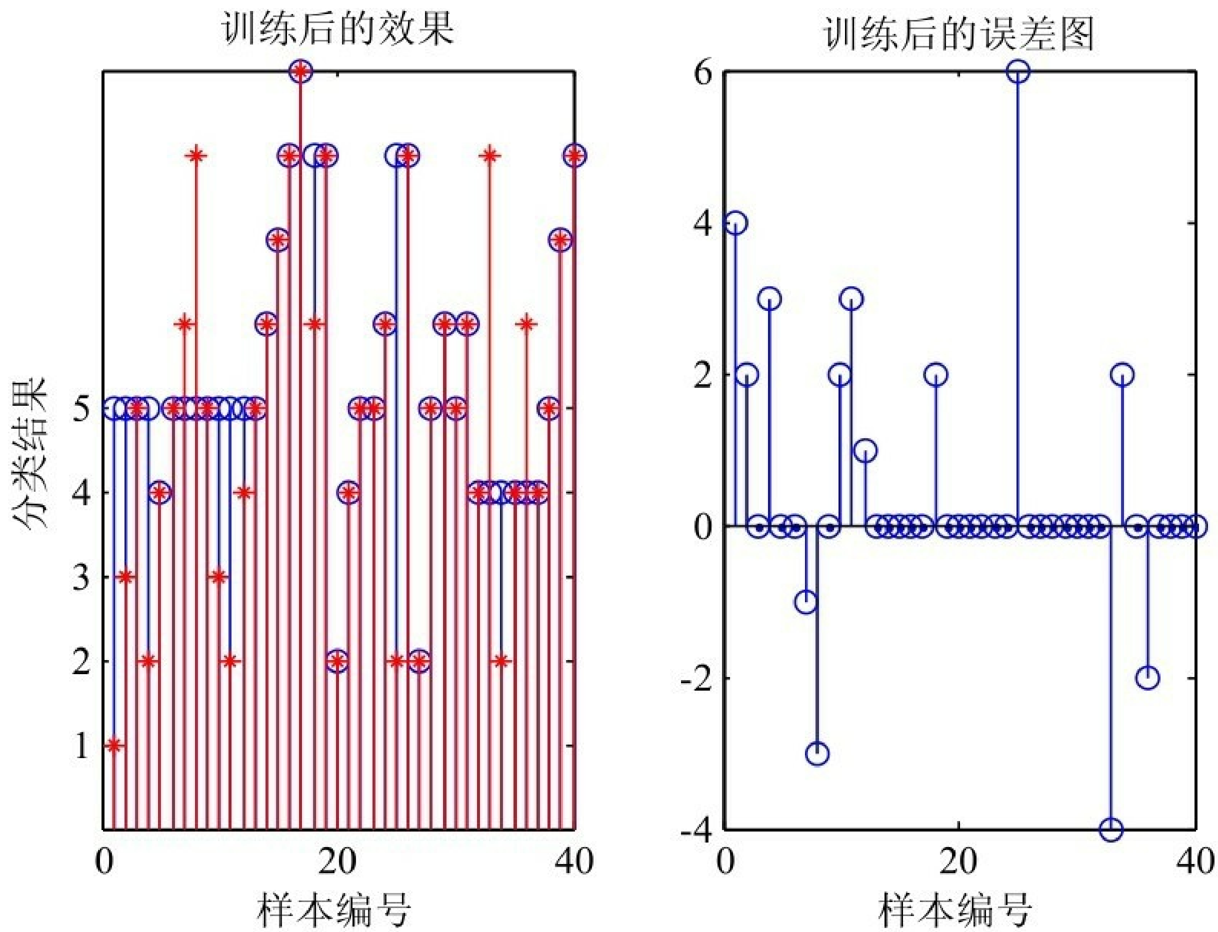


图16-32 训练后训练数据网络的分类效果

预测测试数据的分类效果图如图16-33所示。

由图16-32和图16-33可知，在训练后，将训练数据作为输入代入已经训练好的网络中，有 13 个样品判断错误，并且用预测样本进行试验的时候，有 15 个样本故障类型判断错误，说明概率神经网络对样品测试数据的效果变差，需要用更多的样本对网络进行训练。

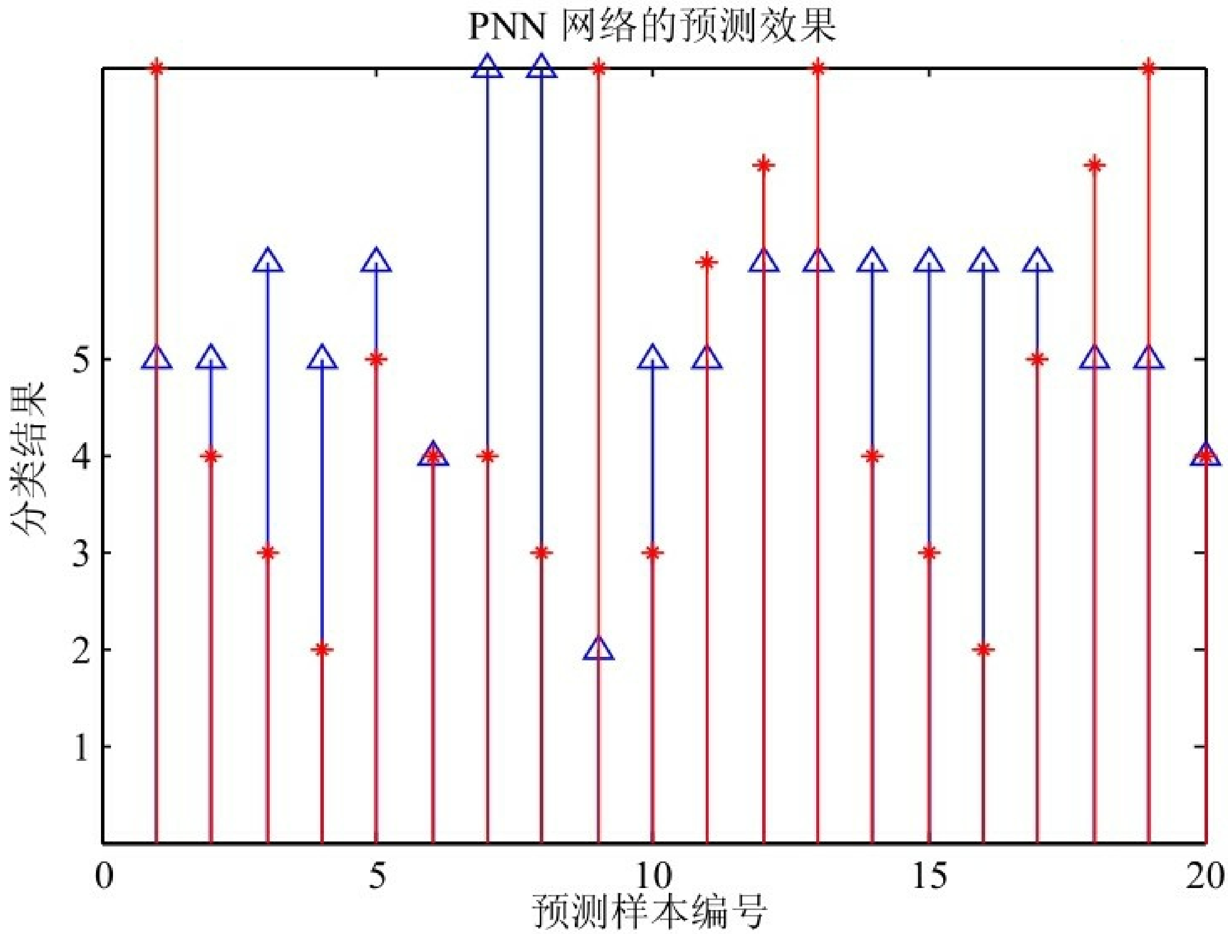


图16-33 预测测试数据的分类效果图

16.6 本章小结

本章利用典型案例，分别讲解了BP神经网络、PID神经网络控制、模糊神经网络和概率神经网络的各种应用。对应前面几章没有涉及的遗传算法等内容，也做了简单介绍，并举例说明其应用。

附录

说明：本文档中所列出的函数适用于MATLAB 5.3 以上版本，为了简明起见，只列出了函数名，若需要进一步的说明，请参阅MATLAB的帮助文档。

1.网络创建函数

newp 创建感知器网络 newlind设计一线性层

newlin 创建一线性层 newff 创建一前馈BP网络

newcf 创建一多层前馈BP网络 newfftd 创建一前馈输入延迟BP网络

newrb 设计一径向基网络 newrbe设计一严格的径向基网络

newgrnn 设计一广义回归神经网络 newpnn 设计一概率神经网络

newc 创建一竞争层 newsom创建一自组织特征映射

newhop 创建一Hopfield递归网络 newelm创建一Elman递归网络

2.网络应用函数

sim仿真一个神经网络 init 初始化一个神经网络

adapt神经网络的自适应化 train训练一个神经网络

3.权函数

dotprod 权函数的点积 ddotprod权函数点积的导数

distEuclidean距离权函数 normprod 规范点积权函数

negdistNegative距离权函数 mandistManhattan距离权函数

linkdist Link距离权函数

4.网络输入函数

netsum网络输入函数的求和　　dnetsum网络输入函数求和的导数

5.传递函数

hardlim硬限幅传递函数　　hardlims 对称硬限幅传递函数

purelin线性传递函数　　tansig正切S型传递函数

logsig对数S型传递函数　　dpurelin 线性传递函数的导数

dtansig 正切S型传递函数的导数　　dlogsig 对数S型传递函数的导数

compet 竞争传递函数　　radbas径向基传递函数

satlins 对称饱和线性传递函数

6.初始化函数

initlay层与层之间的网络初始化函数　　initwb 阈值与权值的初始化函数

initzero 零权/阈值的初始化函数　　initnwNguyen_Widrow层的初始化函数

initconConscience阈值的初始化函数　midpoint中点权值初始化函数

7.性能分析函数

mae 均值绝对误差性能分析函数　mse 均方差性能分析函数

msereg 均方差w/reg性能分析函数　　dmse 均方差性能分析函数的导数

dmsereg 均方差w/reg 性能分析函数的导数

8.学习函数

learnp 感知器学习函数　　learnpn 标准感知器学习函数

learnwhWidrow_Hoff学习规则　　learngdBP学习规则

learngdm带动量项的BP学习规则　　learnkKohonen权学习函数

learnconConscience阈值学习函数　　learnsom自组织映射权学习函数

9.自适应函数

adaptwb 网络权与阈值的自适应函数

10.训练函数

trainwb网络权与阈值的训练函数　　　traingd　梯度下降的BP算法训练函数

traingdm 梯度下降w/动量的BP 算法训练函数

traingda 梯度下降w/自适应lr 的BP 算法训练函数

traingdx 梯度下降w/动量和自适应lr的BP 算法训练函数

trainlm Levenberg_Marquardt 的BP 算法训练函数

trainwbl 每个训练周期用一个权值矢量或偏差矢量的训练函数

11.分析函数

maxlinlr线性学习层的最大学习率　　errsurf 误差曲面

12.绘图函数

plotes 绘制误差曲面

plotep 绘制权和阈值在误差曲面上的位置

plotsom 绘制自组织映射图

13.符号变换函数

ind2vec 转换下标成为矢量　　　vec2ind转换矢量成为下标矢量

14.拓扑函数

gridtop 网络层拓扑函数　　　　hextop 六角层拓扑函数

randtop 随机层拓扑函数

参考文献

[1] MathWorks,Inc.MATLAB®version 8.1(R2013a)，User’s Guide，2013.

[2] 张德丰.MATLAB神经网络编程.北京：化学工业出版社，2011.

[3] 韩力群.人工神经网络理论、设计及应用.北京：化学工业出版社，2007.

[4] 陈明.MATLAB 神经网络原理与实例精解.北京：清华大学出版社，2013.

[5] 王小川.MATLAB 神经网络43 个案例分析.北京：北京航空航天大学出版社，2013.

[6] 傅荟璇.MATLAB 神经网络应用设计.北京：机械工业出版社，2012.

[7] MathWorks,Inc.MATLAB®version 7.8(R2009a)，User’s Guide，2009.

[8] MathWorks,Inc.MATLAB®version 7.13(R2011b)，User’s Guide，2011.

[9] 丛爽.面向 MATLAB 工具箱的神经网络理论与应用.合肥：中国科学技术大学出版社，2009.

[10] 朱凯.精通MATLAB 神经网络.北京：电子工业出版社，2010.

[11] 葛哲学.神经网络理论与MATLAB R2007 实现.北京：

电子工业出版社，2007.

[12] 周润景等.基于MATLAB 与fuzzyTECH的模糊与神经网络设计.北京：电子工业出版社，2010.

[13] 张德丰.MATLAB 神经网络应用设计.北京：机械工业出版社，2012.

[14] [美] Edward B.Magrab.MATLAB原理与工程应用.高会生，李新叶，胡智奇，等译.北京：电子工业出版社，2002.

[15] G.F.Franklin，J.D.Powell，A.Emami-Naeini，Feedback Control of Dynamic Systems， Prentice Hall，New Jersey，2002.

[16] 刘金琨.机械系统RBF神经网络控制：设计、分析及MATLAB 仿真.北京：清华大学出版社，2013.